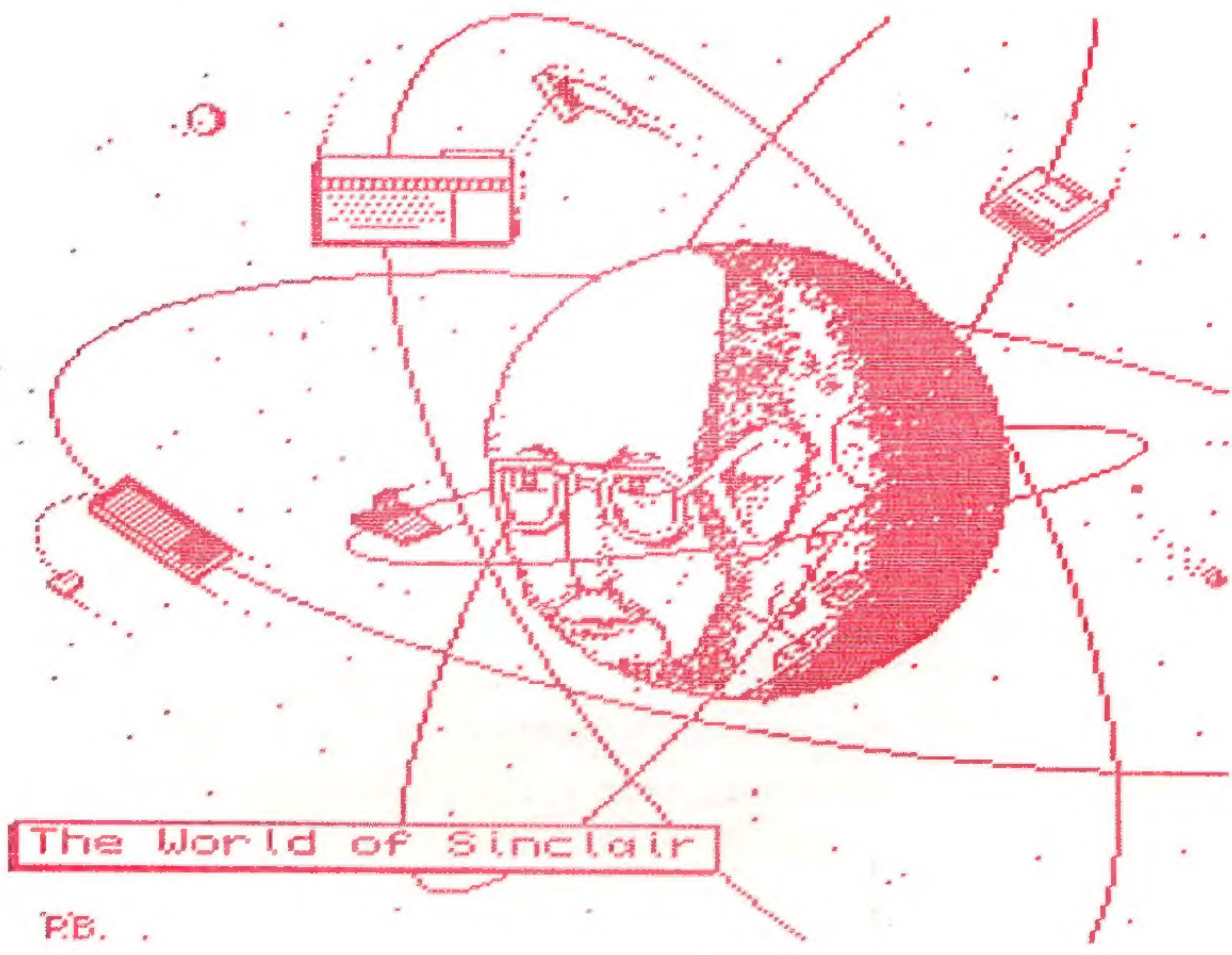
SEPT/OCT 86
VOL.2 NO.6
$3.00

BULK RATE
U.S. POSTAGE
PAID
COLTON, OR 97017
PERMIT #51

TIME DESIGNS™ MAGAZINE

FOR ALL TIMEX AND SINCLAIR COMPUTERS



SEPTEMBER/OCTOBER 86
VOL.2 NO.6

TIME DESIGNS MAGAZINE

FOR ALL TIMEX AND SINCLAIR COMPUTERS

TIME DESIGNS MAGAZINE CO.
29722 Hult Rd.•Colton, Oregon 97017
(503) 824-2658

TIME DESIGNS MAGAZINE is published bi-monthly and is Copyright © 1986 by the **Time Designs Magazine Company,** Colton, Oregon 97017. All rights reserved.

Editor: Tim Woods

Assistant Editor: Stephanie Woods

Editorial Assistant/Production: D.L. Woods

Photography:
(unless otherwise noted): Thomas Judd
Printing by: Toad'l Litho Printing and Comp., Oregon City, Oregon 97045

SUBSCRIPTIONS: $15 a year for six issues (US funds only). No extra charge to Canadian subscribers. All other countries please write for information on air mail rates.

CUSTOMER SERVICE: Customer satisfaction is our goal. For subscription service problems please write or call TIME DESIGNS.

CHANGE OF ADDRESS: Write or call to prevent delay of sevice,

COMMENTS by the editor

Commodore...Soon to join our ranks?

It's been a downhill slide ever since Jack saw the greener grass over at Atari, packed his bags and left Commodore, the company he saw rise to prominence in just three years. Things are bad at Commodore. Real bad. Experts say the the company is losing nearly 120 million a year--that figures out to be around 3 million a week!

If Commodore does bail out, one can only wonder what's in store for countless enthusiasts world-wide. Will the small company and cottage industry be the main source of support, just like our own community emerged two years ago? Will the slick magazines bail out (just like SYNC and TIMEX/SINCLAIR USER did)? A lot of after-market software and hardware houses are counting on the health of Commodore for their very existence.

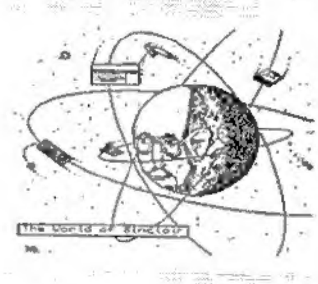
What does all of this mean to you and me? Actually, I want to use Commodore as a "vehicle" for my annual lecture on SUPPORT. Where would we be without our disk drives, printer interfaces, Spectrum Emulators; not to mention some really great home-brew software and various publications? We would have been dead in the water months ago.

Will our supply of computer "goodies" last. I believe so...but it will require everyones participation and SUPPORT. Unless we send for that great-looking new program or board for our Sinclair, or even just respond to catalog offers that we read in newsletters and mags--we won't have Aerco, E. Arthur Brown, Zebra Systems, Novelsoft, Weymil Corp, Curry, Knighted--so forth and so on. If you have been thinking about a new purchase...now is the time...tomorrow maybe a little too late.

I always wonder as I'm "pasting up" the ads in TIME DESIGNS, just how many readers actually pay attention to them. Some company has paid us good money for a spot in the magazine. This helps offset our production costs. Please SUPPORT our dear advertisers, they SUPPORT us.

Well, another lecture has come to a close. I'll get down off my soapbox now. Enjoy this issue of TDM, and get ready for the next one--our Second Anniversary Issue. And what a celebration that one's going to be!

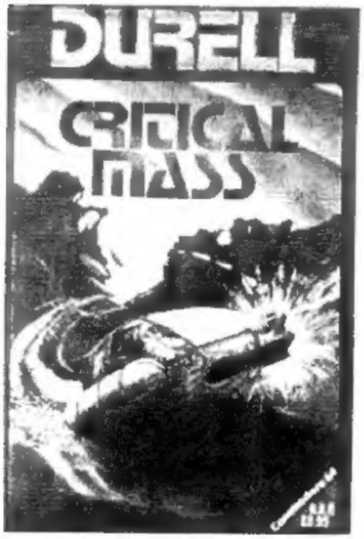
Tim Woods
"the editor"

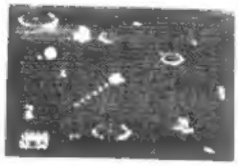


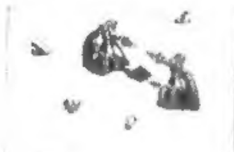
on the cover:

Regular columnist, Paul Bingham is the cover artist this issue, with his rendering of 'The World of Sinclair'. Included is the ZX80, ZX81 + 16K, Spectrum (in 2068 clothing) + ZX Printer, and a QL with Mouse, in Geosyncronous orbit about Uncle Clive. Paul's artwork was done using QL Paint (an updated version of GraphiQL).

★★★★★★★★★★★★★ NEW FOR YOUR T/S 2068 ★★★★★★★★★★★★★



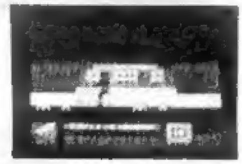




CRITICAL MASS (Arcade)

An outlying system of the Terra Federation has set up an advanced anti-matter conversion plant on a centrally positioned asteroid to supply the local colonists with energy. A surprise attack by alien forces has successfully overcome the asteroid's defences and the aliens are now threatening to self-destruct the power plant unless the colonists offer an unconditional surrender. The self-destruction process would effectively turn the power plant into a massive black hole that would wipe out the entire planetary system along with a number of nearby stars. Unconditional surrender offers an equally horrific prospect. Your mission is to infiltrate the enemy position and disable the anti-matter plant before the aliens achieve CRITICAL MASS.

CATALOG #1026 $15.95



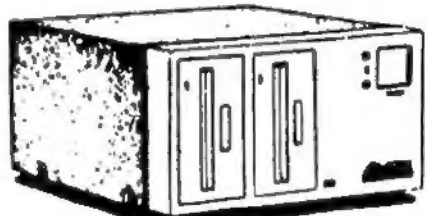
SABOTEUR (Martial Arts Arcade Adventure)

You are a highly skilled mercenary trained in the martial arts. You are employed to infiltrate a central security building which is disguised as a warehouse. You must steal a disk that contains the names of all the rebel leaders before its information is sent to the outlying security stations. You are working against the clock, both in getting to the disk, and in making your escape. You must enter the building from the sea by rubber dinghy, and will then only be able to leave by the helicopter on the roof. You will have to find your way around the warehouse, making use of the guards' own weapon supplies in your struggle through overwhelming odds.

CATALOG #1098 $15.95

★★★★★★★ DUAL DISC DRIVES & INTERFACE FOR QL --$250.00 ★★★★★★★

DUAL DISK DRIVES & INTERFACE FOR YOUR QL COMPUTER .... Now you can cut the rising costs of data storage on your QL in half (at least). These 3" drives are answer to your prayers........top quality and they hold about 160K per side (the "hard" cased floppies can be flipped over) for additional 160K of storage space! And, don't forget the great convienience (not to mention the super reliability of these disks) of having a second disk drive on line! All this...including and we'll even put the WP,DB,SS, and BG programs on one of the two disks supplied with the drives -- ready to run -- and our regular disk drive interface ALL INCLUDED..................NOW ONLY $249.95 !



DISK DRIVES

ADD 5.00 S&H

3" DISKS - (for above) - Encased in hard plastic with a sliding metal "door" to keep dust (& fingerprints) from getting onto the disks. The metal door opens automatically when the disk is put into the drive. BULLETIN - A NEW FACTORY JUST OPENED UP IN HONG KONG TO PROCUDE THE 3" DISKS - WE BELIEVE A PRICE REDUCTION FOR THESE DISKS WILL FOLLOW SOON! $4.50 each or 10 Disks for $43.50

COMPARE OUR --- DELIVERED --- PRICE

WRITE OR CALL FOR OUR FREE CATALOG ---ALL NEW PRICES---MANY NEW ITEMS

KNIGHTED COMPUTERS

707 Highland Street FULTON, NY 13069

(315) 593-8219

LETTERS

To the Editor,

I really enjoy your magazine and please keep up the good work! I have a question...do you know of any BBS program for the 2068 (using the 2050 modem)?

I have included two graphic printouts that I made using a program that I call "draw". Hope you enjoy them as much as I did making them.

Truly yours,

Clifton Tiddle
Diamond Bar, CA

Editor: Clifton, your in luck. I just talked to Eben Brown on the phone, of E. Arthur Brown Co. (3404 Pawnee Drive, Alexandria, MN 56308, 612/762-8847). He has a new BBS program that will be featured in his next catalog. It's called CASBOARD 2068, and is available on either cassette or A&J wafer. The Load and Save commands are in BASIC, so the program could be easily converted to any disk drive system. CASBOARD, was programmed by Kurt Casby, best known for his LOADER series of programs. Preliminary specifications sound great...and should be a lot of fun setting up a small home-based BBS. Price is $19.95 plus $1.95 for postage.

I did enjoy your drawings...and I've printed them here for others to admire.

To the Editor and the Readers:

Can anyone come up with a simple program for my T/S 2068 which will print the characters immediately on the printer instead of on the screen? With this capability I would like to use the computer and its printer without the TV/monitor to do simple computations. Can anyone help?

Sincerely,

Michael J Nowak
San Diego, CA

Editor: I'll forward any possible solutions on to Mr. Nowak that we receive, as well as publish them in an upcoming issue. I assume that the reference to a "printer" means the Timex 2040 thermal printer. C'mon programmers!

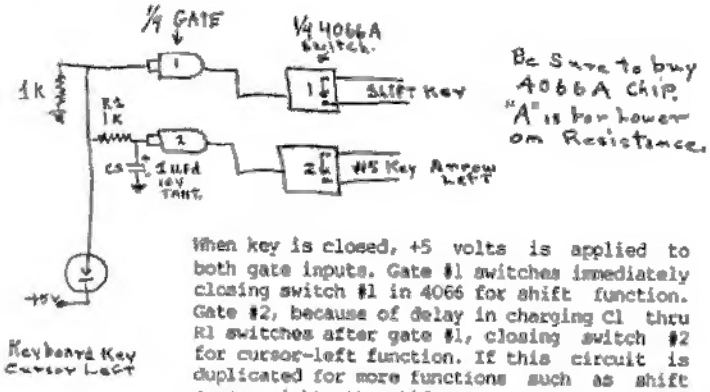
Dear Tim,

Thank you for showing interest in my little gizmo. I have written to other publications, but no one even answered to say the were not interested. You have my permission to print my address. I will be glad to correspond with anyone who needs more information on adding automatic two-key entry from a one-key closure (to an external keyboard like the TI 994/A surplus keyboard).

Sincerely,

D. (Sandy) Rea
Rt. 1 Box 18
Priest River, ID 83856

Editor: I appreciate you sharing your circuit diagram with us.



When key is closed, +5 volts is applied to both gate inputs. Gate #1 switches immediately closing switch #1 in 4066 for shift function. Gate #2, because of delay in charging C1 thru R1 switches after gate #1, closing switch #2 for cursor-left function. If this circuit is duplicated for more functions such as shift cursor-right, the shift key sections of the 4066 may be tied in parallel, as many sections of the 4066 may be done this way, as desired. The gates' chip can be any OR function chip, TTL, CMOS or LS.

Tim:

I remember some time ago in TDM that you asked readers to send some simple programs for publication, so enclosed is one for the 2068, which you may use if you wish. As you can see, this calculates the number of days between any 2 dates (I think the limits are 1900 to 2010) and accounts for leap years, also.

By the way, in the Jan/Feb 86 issue, page 3, you published a T/S 1000 "bubble sort" for a reader. I could not make the program work without deleting line 230 (LET N=1).

Leo Schroeder
Billings, MT

```
  1 REM DAY BETWEEN DATES
      POKE GIVES KEY BEEP
  10 POKE 23609,100: GO TO 900
  100 LET Q=Y+(M<3): LET J=INT (2
75*M/9)-INT (7*Q+7)/4+367*Y+D-IN
T ((INT (Q/100)+1)*3/4): RETURN
  924 INPUT "FIRST DATE? MO-DAY-Y
R";M,D,Y: PRINT "FIRST DATE ";M;
" ";D;" ";Y
  925 GO SUB 100: LET J1=J
  926 INPUT "LAST DATE? MO-DAY-YR
";M,D,Y: PRINT "LAST DATE ";M;"
";D;" ";Y
  928 GO SUB 100: PRINT "DAYS BET
WEEN DATES ";J-J1
```

Sir Clive's Confessions

Entrepeneur extraordinaire, Sir Clive Sinclair, has ended several months of silence, following the sale of major interests in his company that pioneered low-cost home computers, to Amstrad Consumer Electronics PLC. Last week, Sir Clive came out of seclusion and spoke with the British press. The following comments were taken from two separate interviews relating to questions posed about Sinclair microcomputers.

Sir Clive on the Spectrum-

"The Spectrum was and still is an enormous success although it is showing it's age a bit. I was rather surprised to see it turn out to be a games machine...we really knew very little about that side of the market."

What about the QL?

"I think the QL was an interesting idea...a sophisticated machine, but in the end it didn't work out very well, as we had originally anticipated. The market for a 68000-based micro wasn't as big as research led us to believe. The QL had teething troubles early on. The truth was, that when the project came up, that later became the QL...I wanted to do the whole thing on the Z80 microprocessor, but most of the engineers and Nigel [Nigel Searle, former Sinclair Marketing Director] wanted to do it on the 68000. I couldn't see the point of that because it seemed to me you were paying a lot of money for the chip and I couldn't see what you were going to be able to do on it that you couldn't already do on the Z80. Sure it was a bit faster in principle...but it wasn't that in practice. Looking back there was no need to go for 68000 technology. We just haven't found a way to use the 68000 that gives any extra benefit to the customer."

Sir Clive on the Microdrives-

"The bad press the Microdrives received was unfounded, I'd defend them absoulutely. I think they were a marvelous approach to low cost mass storage. Their technology and application should be studied further."

What about the Pandora?

"I want to go ahead with the Pandora project. It will not be compatible with either the Spectrum or QL, as we have lost all rights to their technology. I think it will be best in a way, as it opens the door for a new and customized operating system. Most of the portable computers available are compromises of one sort or another. To me, a portable computer must be totally portable and no trouble to use."

Amstrad director, Alan Sugar?

"I hope to keep in touch with Alan Sugar, and I like him very much."

Sir Clive's future?

"I am most happy right where I am now..tinkering with new projects for the future. To be perfectly honest, I have never felt comfortable playing the business manager role."

What's In A Name?

Early in 1983, advertisements for RAMEX INTERNATIONAL appeared in the national Sinclair magazine SYNC. Ramex, of Utica, Michigan, sold external ZX81 keyboards, RAM packs, among other items. With the advent of the 2068, the company expanded further, under the direction of general manager, Scott Duncan. They obtained the exclusive marketing rights to TASWORD TWO word processor from Tasman Software in Great Britain. Later on, after Timex bailed out of the U.S. computer market, Ramex imported a Spectrum disk drive interface for the T/S 2068 and coupled it with quad drives as a package...it was called the "Millennia K". An "overkill" with the quad drives later led to an Amdek 3" disk system.

Then in February of 1986, Ramex announced that they were no longer supporting Timex computers or their disk drive system, but instead opted to carry the American version of the QL from Sinclair Research plus peripherals and software. They changed the name Ramex International to FOUNDATION SYSTEMS. About the same time, they moved from their original facilities to an address in Washington, Michigan. A new public relations manager wrote a review on the QL, which appeared in the February issue of Computer Shopper.

When A+ Computer Response of Keene, New Hampshire, took over distribution of the QL in the U.S., Foundation Systems became a fully authorized dealer.

In June and July of this year, several readers of TDM wrote, requesting assistance in contacting Foundation Systems, because QLs they had sent for were not being delivered. What TDM learned was that the distributor, A+ Computer Response was receiving similar complaints, and that Foundation's phone number had been disconnected. A spokesperson for A+ informed TDM that Foundation was no longer an authorized QL dealer.

By surprise, in August, TDM was informed that once more, another company had surfaced using the same Washington, Michigan address...it is called MATRIX TECHNOLOGIES. The company took out a small ad in the September 1986 issue of Family Computing Magazine. The new product? IBM PC clones for $399.

NOVELSOFT Emerges As Premier TS Software House

If one were to imagine what the ideal Timex/Sinclair software company would be like, some attributes might include a full time office, open for customer's questions and support, professional products with "complete" documentation at a fair price and prompt order processing. NOVELSOFT of Toronto, Canada, may come close to filling the bill.

The reviews are out, and the word is spreading fast about recent software releases from Novelsoft; TIMACHINE, quite possibly the best BASIC Compiler ever for Sinclair computers; ARTWORX Version 1.1, a sophisticated graphics package; and a brand new release called THE WORX!, which is a collection of useful mini-programs. All of these releases are on their way to achieving "hit" status in the T/S community...a small accomplishment that would blush next to the giant software houses, but a respectful one never the less.

According to Novelsoft, Senior Partner, David Ridge, the company was started to promote their programs in Great Britain. They have had some interested U.K. software publishers, but the current situation in England with Amstrad and the Spectrum, has put a halt to most major investments. The whole industry is waiting to see what will happen next.

Novelsoft has generously included a version of their popular programs on one side of the tape for the Timex/Sinclair 2068. The other side has a Spectrum version of the program.

Product/Dealer News

Sinclair telecomputing experts Ed Grey and Dave Clifford (G & C Computer Products), based in Southern California, have officially announced the release of SPECTERM-64 terminal software and the Z-SI/O card, an RS-232C interface for the T/S 2068. The Specterm-64 software will operate on a Spectrum-emulated T/S 2068. A stock 2068 version is planned for a later release. The terminal software includes a true 64 column display, uses XMODEM protocol for file transfer, will transfer all control characters including ESC, has a 35K+ buffer, and is fully compatible with the T/S 2050 modem and the Sinclair Microdrives. Specterm-64 comes with extensive documentation, and a special version configured to run the Z-SI/O card. The card was designed and manufactured by Dave Clifford, who also developed the Z-LINK Spectrum interface in 1985. Z-SI/O includes a standard RS-232 connector (DB 25 pin), and a full buss feed-through. It will drive a wide range of peripherals, including any 300 and 1200 baud modem (including Hayes compatibles) with the 2068. Note: Specterm-64 can be overlayed to run almost any RS-232 I/F currently available for the T/S 2068, including the circuit featured in the March/April 1986 issue of TDM. Specterm-64 also has built in 1200 baud compatible routines. Price for Specterm-64 is $30.00 plus $2.00 S&H in U.S. (Canada add $2-U.S.funds). The Z-SI/O card is $75.00 plus $3.50 S&H (Canada add $2). Additional information can be obtained by writing Ed Grey or Dave Clifford at: PO Box 2186, Inglewood, CA 90305, (213) 759-7406 or 516-6648.

Another good value for your T/S modem-ing dollar, is the LOADER V software package by Kurt Casby (25 Battle Creek Court, St. Paul, MN 55119). It is an enhancement for the 2068, 2050 modem and MTERM (Smart II) terminal software. Loader V is the final suite in the "Loader" series previously offered by Mr. Casby. Loader V features: An additional 20 number dialing directory, an auto-repeating dialer, capability to Load Mterm buffer with any standard "Bytes" file, Loads text files created with either TASWORD II or MSCRIPT into Mterm's buffer, an XMODEM protocol, among several other user-friendly features. The program on cassette with complete documentation is priced at $9.95.

Robert C. Fischer, producer of PRO/FILE EXTENSIONS, T/S GRADER, WORD PUZZLER, and WORD GAMES, has changed his address and can now be found at Rt 2, Arizona St., Emerson, GA 30137.

QL SCREEN DUMP is a utility program that allows the user to dump items produced on the screen in SuperBASIC, to any Epson-compatible printer. QL Screen Dump is written in fast, compact machine code and reportedly takes up less than 3/4K RAM. The program is available for $24.95 from E-Z KEY, Suite 75, 711 Southern Artery, Quincy, MA 02169.

The English Micro Connection of Newport, Rhode Island, closed it doors for good on August 5th, due to some "serious health problems". EMC owner and operator, Bob Dyl was an early supporter of TIME DESIGNS, and gave TDM several news items of Sinclair computing in Great Britain, obtained from several trips that Bob made to England. The editorial staff of TDM wish Bob a speedy recovery and best wishes for the future.

Knighted Computers, 707 Highland St., Fulton, NY 13069, (315) 593-8219 has obtained some stock and items as a result of the closure of EMC. For information and prices on some interesting QL goodies, write to either Ray or Joe at Knighted.

Stan Lemke, a regular columnist for TDM, and owner of Lemke Software Development (2144 White Oak, Wichita, KS 67207), has done it again. His new program, COLOSSUS, looks like a winner. The program is a graphics banner designer package that allows the user to create a banner 32 screens long, with a variety of font styles/sizes, and add low-resolution graphics on, over and around the banner text. There are also extensive editing features. Printing is to either 2040 or a full-size printer, with modifications by the user for specific printer/interface combinations. A bonus feature of Colossus is a "movie animation" function, that flips a total of 32 screens at the rate of four screens per second for an interesting effect. The program is available on cassette, with full documentation, and a sample animation file, for $19.95 (postage included).

HIGH RESOLUTION programming for ZX81-based micros, is the trend, up at Fred Nachbaur's workshop (address: C-12 Mtn. Stn. Group Box, Nelson, B.C., V1L 5P1 Canada). We're not sure exactly how he does it, but we do know it takes a lot of memory. Fred's latest offering is a high-res maze adventure game for the Timex 1500 with either a 16K RAM pack or 8K Hunter Board (purchasers must specify which version). A later version for the ZX81 and T/S 1000 will be released. DUNGEON OF YMIR, is 100% machine code, with monsters and multi-levels. Price is $24.95. Other hi-res programs are available.

Users Group Update

T/S User Group Correspondents: Send us your group's address and we will list it in an upcoming issue. We will also print announcements, special events and User Group news (if it's brief).

Anyone interested in forming a T/S User Club in the Leesburg area of (Central) Florida, should contact Warren Fricke, 225A Highland Dr, MFL, Leesburg, FL 32788 or phone (904) 589-2729.

Timex-Sinclair User Club, c/o Mr. Richard K. Norek, 188 St. Felix Ave.k, Cheektowaga, NY 14227.

Timex-Sinclair User Group, 1545 Alta Vista Drive, Apt. 1402, Ottawa, Ontario, Canada K1G 3P4.

Over 45 guests attended the Grand Opening and Open House at the new facilities of Time Designs Magazine on August 30. Attendees included some members of CATS and PATS Users groups of Oregon, as well as a number of subscribers from the Northwest. Highlights included a QL demonstration by TDM writer, Mike de Sosa; preview of LIGHT SHOW 2000 (a program featured in this issue) by the author Michael Carver; Sinclair merchandise was displayed and sold by RMG Enterprises; and there were door prizes and refreshments. The day was enjoyed by all those who came. Tim Woods, Editor of TDM, announced that the Open House would be an annual event.

LIGHT SHOW

Revox (a manufacturer of "Top-of-the -Line" audio equipment) recently introduced a cassette deck with an RS232 port, allowing control of the unit via a computer. For a mere $1,400.00, this cassette deck could be yours. LIGHT SHOW 2000 will turn the tables on the Revox for about 1/100th of the cost. LS 2000 is a program which will allow your cassette deck (or any other musical source) to control your computer.

LS 2000 will poll the ear port of your TS 2068 and decipher any pulse detected into one of four tonal groups (the shorter the pulse, the higher the note). Depending on the tone detected, a corresponding color pattern will be displayed on the screen. The user has control of the tonal groups, colors, duration of display and the speed at which the tones will be read. This flexibility allows one to "view" the same piece of music in a multitude of ways, or to tailor the program to a certain musical selection. LS 2000 comes with one preset Set-Up to "display" music and provides for four user-defined set-ups.

LS 2000 HOOK-UP

To use LS 2000, some means of providing a musical source to the computer must be used. There are several ways of accomplishing this, some more flexible and preferable than others. The simplest means is to hook up a wire directly from the speaker of a stereo system to the earphone jack of the TS 2068. (IMPORTANT: Do not hook up more than one channel of a stereo system as this may damage the stereo amplifier.) This can be done by running speaker wire from the rear connectors of a speaker (or from the speaker output of your stereo) to your computer. Do not leave the speaker disconnected from the amplifier. A phone jack can be attached to the ends of the speaker wire and plugged directly into the computer's ear jack. Or alligator clips can be used to make a connection to your computer patch cords. The drawback of this approach is lack of control over the signal going into the computer. If the music is played at a low volume, the signal may be too low. Conversly, "Heavy Metal" from a 200+ watt system at full blast may cause your 2068 to become light dust.

If the tape recorder you use with the 2068 will play through the earphone jack while in record mode, it can be used to feed the sound source into the computer. A similar wire will be needed to plug into the microphone jack of the recorder, as previously discussed. Run a patch cord from the earphone jack of the recorder to the earphone jack of your computer. Place a tape in the recorder and set it to record. If you opt for this method and plan to play your music at medium to high volume, I would suggest placing an attenuater in-line between the microphone input and the speaker wire from the stereo. (An attenuater may be obtained from Radio Shack for $1.99 -- Part #274-300. This part has an RCA jack for input and a regular mic/earphone jack for output.) This will help prevent distortion and possible overloading of your recorder. The preferable choice is to use Radio Shack's mini-amplifier ($11.95 -- Part #277-1008). The hook-up is the same as with the tape recorder. This method will allow control of the signal volume going into the computer. Once again, use an attenuater, if the musical source is to be played at any volume. If LS 2000 does not respond to inpute while using the attenuater, the attenuater should be removed. (NOTE: This mini-amplifier can be used to boost the output of computer tapes you may have difficulty loading. It also can be used to amplify BEEP output from your computer.)

A "walkman" type cassette player can also be used to supply music to the computer if it has two headphone jacks. Use one of the jacks to run a patch cord to the computer.

USING LS 2000

Upon running LS 2000 you will be presented with a main menu (see example 1). "ENTER LIGHT SHOW" (Option 0) will pulse color patterns on the screen based on the input through the ear port. (NOTE: To return to the main menu while in this mode, press the "q" Key.) Option 1, "SYSTEM SET-UP", will provide a second menu allowing the user to select 1 of 5 permutations of LS 2000 (see example 2). The current set-up is highlighted via BRIGHT. (NOTE: If any of the parameters are changed, no current set-up is shown.) This menu also allows viewing of the parameters for any compiled set-up (Option V). Option S will define a set-up based on the current setting (i.e.; mode, colors, tone, pulse, tempo). The user is prompted to choose a number to be compiled (2-5) and for a name. When this new definition is compiled, it then becomes the current set-up.

From the main menu, the user can create different set-ups or setting. By changing any of the various options (2-6), LS 2000 can be customized to any musical input or user preference. The best way to learn what each option does is to experiment. After changing an option, one can view the results by "Entering Light

PARTS LISTS

Radio Shack Part #	Description	Price
42-2370	Patch cord w/RCA phono jack to stripped wire 36 in.	$ 1.69
42-2371	" " 72 in.	$ 1.89
42-2372	" " 144 in.	$ 2.19

NOTE: The above to be used with Radio Shack Attenuater. Choose the length to fit your needs. Speaker wire may be pig-tailed to stripped ends if needed.

274-300	RCA Phono jack to 1/8" signal reducer (attenuater)	$ 1.99
274-287	Red 2 conductor 1/8" phone plug (2 per package)	$ 1.29
277-1008	Mini Audio Amplifier w/speaker	$11.95

```
ENTER LIGHT SHOW...................0
SYSTEM SET-UP......................1
SET MODE...........................2
SET COLORS.........................3
TONE CONTROL.......................4
SET PULSE..........................5
SET TEMPO..........................6
SAVE/LOAD SET-UPS..................7

"q" Returns from Light Show
```

example 1

2000 by Michael E. Carver

Show". Each setting option is provided with prompts and explanations from within the program. (See Sample Set-ups for examples.) Depending on the type of music or the quality of the input (dynamic range), one may need to retune the tone control. Tone 0 is the highest tone range, Tone 3 the lowest. The number assigned to a tone is the upper limit at which LS 2000 will produce a pulse.

The Save/Load option allows the saving of favorite compiled setting to tape for later retrieval.

```
DEFAULT SETTING.................1
Laid-Back.......................2
Speed-O.........................3
SETTING NOT DEFINED.............4
Ripple..........................5
COMPILE CURRENT SET-UP..........S
VIEW SET-UP.....................V
RETURN TO MAIN MENU.............M
```

example 2

BEHIND THE SCENES

When the TS 2068 is loading a program from tape, it reads through the ear port (port FEh) the signals recorded on the tape. The data needed to send the program is tored in bit 6. If the bit is set ("1") the frequency of the signal is 1020hz, if it is not set ("0"), the frequency is 2040hz. The frequency is determined by the length of the pulse detected. Port FEh also uses bits 4-0 to poll the keyboard. By sending out this port, BORDER colors can be controlled (bits 2-0) or a BEEP can be triggered through bit 4. When a program is sent to tape, bit 3 of port FEh is used. The threshold of the ear port is 23khz, with the input being 4-10 volts p-p.

KEYING IN THE LISTING

Carefully key in the BASIC listing. After you have typed in the program, SAVE the listing to tape before running the machine code loading routine. To load the machine code portion, ENTER as a direct command [RUN 9000]. This portion of the program will POKE the machine code into its proper address. It also checks for various typing errors and will provide instructions in case an error was detected. After the machine code has been placed in memory, the program will set up the User Graphic "A", delete this portion of the program from the listing, and prompt you to SAVE & VERIFY the completed program along with the compiled code. After VERIFYing, the program will self-run. Go ahead and try it out. NOTE: In Line 3, the A in quotes is typed via GRAPHIC mode [Caps Shift/9] [A] [Capts Shift/9].

SAMPLE SET-UPS

Name	Mode	Tone Limit #0	#1	#2	#3	Timing Course/Fine Tempo	Pulse
Default	1	240	208	192	160	10/256	1/1
Laid-Back	1	245	208	192	160	10/1	12/256
Speed-O	0	215	208	192	160	2/150	1/1
Pulser	0	215	208	192	170	1/1	10/176
Ripple	1	215	208	192	170	3/100	10/176

BASIC LISTING

Lines	Notes
1-3	Sets up the screen with the patterns to be "pulsed" by LS 2000. NOTE: Even when the screen looks blank, the complete pattern is still on the screen, as INK has been set the same color as the PAPER. The machine code simply sets the ATTRIBUTES to the proper INK color. Check this out by changing the INK color in Line 1 to "7". Add Line 9 STOP. As direct command <GO TO 1>
10	The POKE sets lower case only. This is the USR call for the machine code portion of LS 2000.
100-120	Contains data for line/column placement of LS 2000 graphics.
130	Contains data defining USR Graphic "A". Can be redefined to any character or pattern.
140-160	Sets up and defines variables for LS 2000 BASIC Line 150 Contains data for Default "Set-Up"
1000-1010	Main Menu
2000-2100	System Set-Up Menu
2200-2240	Compiles current parameters as a defined Set-Up
2300-2320	POKEs parameters into Machine Code as current Set-Up. (See Line 2999)
2400-2450	Displays parameters for a compiled Set-Up
2999	Contains addresses of Machine Code which hold parameters for current Set-Up
3000-3050	Option 2 -- Set Mode
3100-3170	Option 3 -- Set Colors
3200-3280	Option 4 -- Tone Control
3300-3390	Option 5 -- Pulse Control & Option 6 -- Tempo Control. NOTE: This subroutine is shared by both Options. Control of Option is decoded by variable A: IF 5 THEN Pulse Control, IF 6 THEN Tempo Control
4000-4200	Save/Load Option. Also allows for verification of SAVE and LOAD without Breaking the program with a ROM Error Report.
8000-8030	Subroutine to skan keyboard for input
9000-9420	Routine to POKE Machine Code into memory
9998-9999	Routines to SAVE and LOAD LS 2000

The author will provide a copy of this program on tape for $4.00 (includes shipping). Please send a check or money order to: Michael E. Carver, 1016 NE Tillamook, Portland, OR 97212. Please specify "Light Show 2000".

LIGHT SHOW 2000

```
   1 BRIGHT 0: BORDER 0: RESTORE : PAPER 0: INK 0: CLS : LET x=2
   2 FOR a=1 TO 69: READ y: IF y=255 THEN  LET x=x+1: NEXT a
   3 PRINT AT x,y;"A": NEXT a
  10 POKE 23658,0: RANDOMIZE USR 45056: INK 9: GO TO 1000
 100 DATA 15,255,12,18,255,15,255,10,13,17,20,255,15,255,8,12,14,16,18,22,255,10,15,20,255,12,14,16,18
 110 DATA 255,7,9,11,13,15,17,19,21,23
 120 DATA 255,12,14,16,18,255,10,15,20,255,8,12,14,16,18,22,255,15,255,10,13,17,20,255,15,255,12,18,255,15
 130 DATA 0,BIN 1000010,BIN 111100,BIN 11000,BIN 11000,BIN 111100,BIN 1000010,0
 140 RESTORE 150: DIM s(5,14): FOR a=1 TO 14: READ b: LET s(1,a)=b: NEXT a: DIM s$(5,31): LET s$(1)="DEFAULT SETTING.................": LET d$=".............................": FOR a=2 TO 5: LET s$(a)=d$: NEXT a
 150 DATA 1,0,240,208,192,160,6,3,5,2,10,0,1,1
 160 LET current=1: DIM b$(32): DIM c$(3)
1000 PAPER 1: BORDER 1: CLS : PRINT AT 0,8; INVERSE 1;"LIGHT SHOW MENU";AT 4,0; INVERSE 0;"ENTER LIGHT SHOW...............0"''"SYSTEM SET-UP.................1"''"SET MODE......................2"''"SET COLORS....................3"''"TONE CONTROL..................4"''"SET PULSE.....................5"''"SET TEMPO.....................6"''"SAVE/LOAD SET-UPS.............7"
1005 PRINT #0;"  ""q"" Returns from Light Show": ON ERR  RESET
1010 GO SUB 8000: GO TO 1010+(990 AND k$="1")-(1009 AND k$="0")+(1990 AND k$="2")+(2090 AND k$="3")+(2190 AND k$="4")+(2290 AND (k$="5" OR k$="6"))+(2990 AND k$="7")
2000 PAPER 2: BORDER 2: CLS : REM system set-up########
2010 INPUT INKEY$: PRINT  INVERSE 1;AT 0,9;"SYSTEM SET-UP"; INVERSE 0;'''''
2020 FOR a=1 TO 5: PRINT (s$(a) AND s(a,1))+("SETTING NOT DEFINED............." AND NOT s(a,1));a'': NEXT a
2030 PRINT "COMPILE CURRENT SET-UP..........S"''"VIEW SET-UP.....................V"''"RETURN TO MAIN MENU.............M"
2040 IF current THEN  PRINT  BRIGHT 1; OVER 1;AT current*2+3,0;b$
2100 GO SUB 8000: GO TO 2100+(100 AND (k$="s" OR k$="S"))+(200 AND (k$>="1" AND k$<="5"))+(300 AND (k$="v" OR k$="V"))-(1100 AND (k$="m" OR k$="M"))
2200 REM compile set-up
2210 PRINT AT 15,0; OVER 1; PAPER 3;b$: PRINT #0;"Set-Up # (2-5)": GO SUB 8000: IF k$<="1" OR k$>="6" THEN  INPUT INKEY$: BEEP .35,10: GO TO 2010
2220 LET k=VAL k$: PRINT AT k*2+3,0; OVER 1; FLASH 1;b$: RESTORE 2999: LET s(k,1)=1: FOR a=2 TO 14: READ b: LET s(k,a)=PEEK b: NEXT a
2230 INPUT "Set-Up Name? "; LINE k$: LET s$(k)=d$: LET s$(k, TO (LEN k$ AND LEN k$<30)+(30 AND LEN k$>=30))=k$
2240 LET current=k: GO TO 2010
2300 REM make set-up current
2310 LET k=VAL INKEY$: IF s(k,1)=0 THEN  PRINT AT k*2+3,0; OVER 1; FLASH 1;b$: BEEP .35,10: FOR a=1 TO 50: NEXT a: INPUT INKEY$: GO TO 2010
2320 RESTORE 2999: LET current=k: FOR a=2 TO 14: READ b: POKE b,s(k,a): NEXT a: GO TO 2010
2400 REM View Set-Up
2410 ON ERR  GO TO 1000: PRINT PAPER 3; OVER 1;AT 17,0;b$;#1;"View Set-Up # (1-5)": GO SUB 8000: IF k$<="0" OR k$>="6" THEN  INPUT INKEY$: BEEP .35,10: GO TO 2010
2420 LET k=VAL INKEY$: IF s(k,1)=0 THEN  PRINT AT k*2+3,0; OVER 1; FLASH 1;b$: FOR a=1 TO 100: NEXT a: BEEP .35,10: INPUT INKEY$: PRINT AT k*2+3,0; OVER 1; FLASH 0;b$: GO TO 2010
2430 CLS : PRINT  INVERSE 1;s$(k);k; INVERSE 0''"MODE ";s(k,2)''"Tone Color and Limit": PRINT  PAPER 0; INK s(k,7);AT 6,3;"TONE 0 ";CHR$ 144;" ";s(k,3), INK s(k,8);" TONE 1 ";CHR$ 144;" ";s(k,4);AT 7,3; INK s(k,9);"TONE 2 ";CHR$ 144;" ";s(k,5), INK s(k,10);" TONE 3 ";CHR$ 144;" ";s(k,6): PAPER 2
2440 PRINT '''"Timing"''TAB 4;"Course"," Fine"'''"Tempo  ";s(k,11)+(256 AND s(k,11)=256);TAB 18;s(k,12)+(256 AND s(k,12)=0)''"Pulse  ";s(k,13)+(256 AND s(k,13)=0);TAB 18;s(k,14)+(256 AND s(k,14)=0)
2450 PRINT #0;"Press any key to return to Menu": GO SUB 8000: GO TO 2000
2999 DATA 45351,45076,45080,45084,45088,45129,45117,45105,45093,45213,45216,45196,45199
3000 REM set mode
3005 ON ERR  GO TO 1000: PAPER 3: BORDER 3: CLS : PRINT TAB 11; INVERSE 1;"SET MODE"
3010 PRINT ''''"Mode 0 -- Tone Pattern will staylit only during ON Period."'''"Mode 1 -- Tone Pattern will staylit until next Tone Pulse."'''"Currently set at "; INVERSE 1;"MODE ";PEEK 45351
3020 PRINT #0; BRIGHT 1;"Enter Desired Mode (0 or 1)      ""M"" for Menu";b$(13 TO )
3030 GO SUB 8000: IF k$<>"0" AND k$<>"1" AND k$<>"M" AND k$<>"m" THEN  GO TO 3030
3040 IF k$="1" OR k$="0" THEN  POKE 45351,VAL k$: LET current=0: INPUT INKEY$: PRINT AT 0,0: GO TO 3010
3050 GO TO 1000
3100 REM set colors
3110 ON ERR  GO TO 1000: PAPER 4: BORDER 4: CLS : PRINT  INVERSE 1;AT 0,10;"SET COLORS"'''
3120 PRINT  BRIGHT 1; INK PEEK 45129; PAPER (4 AND PEEK 45129=0);AT 6,3;"TONE 0 ";CHR$ 144;" ", INK PEEK 45117; PAPER (4 AND PEEK 45117=0);" TONE 1 ";CHR$ 144;" ";AT 7,3; INK PEEK 45105; PAPER (4 AND PEEK 45105=0);"TONE 2 ";CHR$ 144;" ", INK PEEK 45093; PAPER (4 AND PEEK 45093=0);" TONE 3 ";CHR$ 144;" "
3130 INPUT INKEY$: PRINT #0;" Tone # (0-3)   -- ""M"" for Menu"
3140 GO SUB 8000: IF k$<"0" OR k$>"3" AND (k$<>"M" AND k$<>"m") THEN  GO TO 3140
3150 IF k$="m" OR k$="M" THEN  GO TO 1000
3160 LET a=VAL k$: INPUT INKEY$: PRINT #0;"New color for TONE ";k$;"   (0-7)"
3170 GO SUB 8000: IF k$<"0" OR k$>"7" THEN  GO TO 3170
3180 POKE 45000+(129 AND a=0)+(117 AND a=1)+(105 AND a=2)+(93 AND a=3),VAL k$: LET current=0: GO TO 3120
3200 REM tone control
3210 ON ERR  GO TO 1000: PAPER 5: BORDER 5: CLS : PRINT  INVERSE 1;AT 0,9;"TONE CONTROL"''
3220 BRIGHT 1: PAPER 0: PRINT  INK PEEK 45129;AT 6,3;"TONE 0 ";CHR$ 144;" ";: LET c$=STR$ PEEK 45076: PRINT c$;, INK PEEK 45117;" TONE 1 ";CHR$ 144;" ";: LET c$=STR$ PEEK 45080: PRINT c$;AT 7,3; INK PEEK 45105;"TONE 2 ";CHR$ 144;" ";: LET c$=STR$ PEEK 45084: PRINT c$, INK PEEK 45093;" TONE 3 ";CHR$ 144;" ";: LET c$=STR$ PEEK 45088: PRINT c$
3225 PAPER 5: BRIGHT 0
3230 INPUT INKEY$: PRINT #0;" Tone # (0-3)   -- ""M"" for Menu"
3240 GO SUB 8000: IF k$<"0" OR k$>"3" AND (k$<>"M" AND k$<>"m") THEN  GO TO 3240
3250 IF k$="m" OR k$="M" THEN  GO TO 1000
3260 LET a=VAL k$: LET limit1=5-a: LET limit2=(254 AND a=0)+(PEEK 45076-1 AND a=1)+(PEEK 45080-1 AND a=2)+(PEEK 45084-1 AND a=3)
3265 INPUT ;"New limit for TONE "+STR$ a+" ("+STR$ limit1+"-"+STR$ limit2+")?  "; LINE k$
3270 LET k=VAL k$: IF k<limit1 OR k>limit2 THEN  GO TO 3265
3280 POKE 45000+(76 AND a=0)+(80 AND a=1)+(84 AND a=2)+(88 AND a=3),VAL k$: LET current=0: GO TO 3220
3300 REM pulse and tempo control
3310 ON ERR  GO TO 1000: LET a=VAL k$: PAPER 6+(a=6): BORDER 6+(a=6): CLS
3320 PRINT  INVERSE 1;AT 0,9;("PULSE" AND a=5)+("TEMPO" AND a=6)+" CONTROL"
3330 LET course=PEEK (45196+(17 AND a=6)): LET fine=PEEK (45199+(17 AND a=6)): PRINT '''"Current "+("Pulse" AND a=5)+("Tempo" AND a=6)''TAB 4;"Course","Fine"'TAB 6;course+(256 AND course=0);"   ";TAB 17;fine+(256 AND fine=0);"   "
3340 PRINT '''"Course -- Number of times ""Fine""period is repeated"'''"Fine -- Sets timing via nano-   seconds"
3350 PRINT AT 18,0;("Pulse -- Length of time needed  to place Tone pattern on screen." AND a=5)+("Tempo -- Length of pause betweenreading Tones" AND a=6)
3360 PRINT #0;"Set Course or Fine (C or F)?    ""M"" for Menu"
3370 GO SUB 8000: IF k$<>"C" AND k$<>"c" AND k$<>"F" AND k$<>"f" AND k$<>"m" AND k$<>"M" THEN  GO TO 3370
```

```
3380 LET i$=k$: IF i$="m" OR i$=
"M" THEN  GO TO 1000
3390 PRINT  OVER 1; FLASH 1;AT 5
,4+(12 AND i$="4" OR i$="F");"
   "+("  " AND (i$="c" OR i$="C
")); INPUT "Enter # (1-256) ";
LINE k$: LET k=VAL k$: IF k<1 O
R k>254 THEN  GO TO 3390
3400 POKE 45196+(3 AND (i$="4" O
R i$="F"))+(17 AND a=4),k-(256
AND k=256): LET current=0: GO T
O 3320
4000 REM save/load set-ups
4010 ON ERR  GO TO 1000: CLS : P
RINT TAB 7; INVERSE 1;"SAVE/LOA
D SET-UPS";#0; BRIGHT 1;"S = SA
VE    L = LOAD    M = MENU"
4020 GO SUB 8000: IF k$<>"S" AND
 k$<>"s" AND k$<>"L" AND k$<>"l
" AND k$<>"M" AND k$<>"m" THEN
 GO TO 4020
4030 IF k$="S" OR k$="s" THEN  G
O TO 4100
4035 IF k$="M" OR k$="m" THEN  G
O TO 1000
4040 PRINT '''"Ready to LOAD Set-
Ups": INPUT "Load title ?";n$:
PRINT '''"Loading ";n$: PRINT #0
;"Start tape, then press any ke
y.": GO SUB 8000: INPUT INKEY$
4050 IF LEN n$>10 THEN  LET n$=n
$( TO 10)
4060 ON ERR  GO TO 4200: LOAD n$
 DATA a$(): LOAD n$ DATA a(): G
O TO 1000
4100 PRINT '''"Ready to SAVE Set-
Ups": INPUT "SAVE title ?";n$:
LET n$=("Set-Ups" AND n$="")+n$
( TO LEN n$-(LEN n$-10 AND LEN
n$>10)): PRINT AT 6,0;"Saving "
"";n$;""""
4110 SAVE n$ DATA a$(): SAVE n$
DATA a(): PRINT #0;"Verify (Y o
r N)?": GO SUB 8000
4120 IF k$="N" OR k$="n" THEN  G
O TO 1000
4130 INPUT INKEY$: PRINT AT 4,0;
"Verifying  """;n$;""""'''"Rewin
d tape";#0;"Start tape and pres
s any key.": GO SUB 8000
4140 ON ERR  GO TO 4200: VERIFY
n$ DATA a$(): VERIFY n$ DATA a(
): GO TO 1000
4200 CLS : PRINT  FLASH 1;AT 5,7
;"Tape Loading Error": PRINT ''
'TAB 6; INVERSE 1;"Please Attem
pt Again"; INVERSE 0;#0;"Press
any key for Menu": GO SUB 8000:
 GO TO 1000
8000 REM keyboard scan
8010 IF INKEY$<>"" THEN  GO TO 8
010
8020 IF INKEY$="" THEN  GO TO 80
20
8030 LET k$=INKEY$: RETURN
9000 CLEAR 45055: LET a$="": RES
TORE 9100: FOR i=0 TO 36: READ
d$: LET a$=a$+d$: NEXT i
9010 IF LEN a$<>592 THEN  PRINT
 FLASH 1;"Error in Machine Code
 DATA      Lines 9100 - 9136"''
 FLASH 0;"There are too "+("few
 "  AND  LEN a$<592)+("many "
AND LEN a$>592)+"Data Items!"''
"PLEASE CORRECT BEFORE CONTINUI
NG"
9020 LET address=45056: FOR i=1
TO LEN a$-1 STEP 2
9030 POKE address + INT ((i-1)/2
),(CODE a$(i)-(48 AND CODE a$(i
)<58)-(55 AND CODE a$(i)>64))*1
6+CODE a$(i+1)-(48 AND CODE a$(
i+1)<58)-(55 AND CODE a$(i+1)>6
4)
9040 NEXT i
9100 DATA "3A085CFE712001C9"
9101 DATA "16FF15AFDBFECB77"
9102 DATA "20F87AFEF030E9FE"
9103 DATA "D6302CFEA0301CFE"
9104 DATA "78300C110002ED53"
9105 DATA "B05C21A7B0182211"
9106 DATA "0005ED53B05C21C7"
9107 DATA "B01816110003ED53"
9108 DATA "B05C21E7B0180A11"
9109 DATA "0006ED53B05C2107"
9110 DATA "B13A27B1FE012805"
9111 DATA "E5CD7FB0E1CD7FB0"
9112 DATA "189E7EE638577EE6"
9113 DATA "07070707BA2006ED"
9114 DATA "5BB05C18053A8D5C"
9115 DATA "77C9E6F88277C906"
9116 DATA "105E235623E52100"
9117 DATA "5819C50601C50601"
9118 DATA "10FEC110F8C1CD62"
9119 DATA "B0E110E5060CC506"
9120 DATA "0010FEC110F8C90F"
9121 DATA "0151018F014D01CF"
9122 DATA "005301CF014B018F"
9123 DATA "0055010F0249014F"
9124 DATA "0057014F02470130"
9125 DATA "0170016E012E01F2"
9126 DATA "008201AC01EC00B4"
9127 DATA "00F401EA01AA0081"
9128 DATA "0481048104810432"
9129 DATA "0172016C012C0114"
9130 DATA "0194018A010A01F6"
9131 DATA "00B601AB01E80081"
9132 DATA "048104810481044F"
9133 DATA "01F000B001AE01EE"
9134 DATA "00B100F101ED01AD"
9135 DATA "00720032022C026C"
9136 DATA "0081048104810400"
9200 RESTORE 9300: LET tally=0:
FOR i=0 TO 36: READ d: LET tall
y=tally+d: NEXT i: IF tally<>26
767 THEN  PRINT  FLASH 0;"ERROR
"; FLASH 0;" in DATA Lines 9300
 - 9350"''"Please correct and <
RUN 9200>": STOP
9230 RESTORE 9300: LET address=[illegible]
 FOR i=0 TO 36: LET check=
0: FOR j=1 TO 8: LET check=chec
k+PEEK address: LET address=add
ress+1
9240 NEXT j: READ tally: IF chec
k<>tally THEN  PRINT  FLASH 1;"
ERROR"; FLASH 0;" -- in DATA"''"
Check for an Error in Line ";91
00+i''"And then procede by (RUN
9000)": STOP
9250 NEXT i
9300 DATA 759,1268,1431,1036,519
,719,825
9310 DATA 562,759,634,751,1470,1
037,489
9320 DATA 679,1254,528,521,1223,
867,9[illegible]
9330 DATA 812,511,256,289,514,76
8,779
9340 DATA 453,290,546,713,482,83
1,830
9350 DATA 320,399
9400 CLS : PRINT "Machine Code h
as been Loaded     into memory."
''"Press any key to SAVE & VERI
FY  Light Show 2000!": PAUSE 0
9410 INK 9: RESTORE 130: FOR a=U
SR "a" TO USR "a"+7: READ y: PO
KE a,y: NEXT a
[illegible] CLEAR : DELETE 9000,9500
9998 SAVE "ls 2000" LINE 9999: S
AVE "ls 2000"CODE 45056,296: CL
S : PRINT "Rewind Tape and Play
 to Verify": VERIFY "": VERIFY
""CODE : GO TO 140
9999 CLEAR 45055: LOAD "ls 2000"
CODE 45056,296: INK 9: RESTORE
130: FOR a=USR "a" TO USR "a"+7
: READ y: POKE a,y: NEXT a: GO
TO 140
```

Machine Code Listing

ORIGIN B000h (45056d)

[illegible]	OP CODE	LABEL	MNEMONIC	NOTES
B000	3A085C	START	LD A,(5C08h)	;last key pressed
[illegible]	FE71		CP 71h	;was it "q"
B005	2001		JR NZ,LISTEN	;if not continue
[illegible]	C9		[illegible]	;return to BASIC
B008	16FF	LISTEN	LD D,FFh	;set pulse counter
B00A	15	COUNT	DEC D	;count down
B00B	[illegible]		XOR A	;clear A & flags
B00C	[illegible]		IN A,(FEh)	;read ear port
[illegible]	CB77		BIT 6,A	;pulse detected?
[illegible]	20F8		JR NZ,COUNT	;if so keep counting
B012	7A	READ	[illegible] A,D	;A = pulse length
B013	FEF0		CP F0h	;upper limit Tone 0
B015	30E9		JR NC,START	;if too high start over
B017	FED0		CP D0h	;upper limit Tone 1
B019	302C		JR NC,TONE0	
B01B	FEC0		CP C0h	;upper limit Tone 2
B01D	301C		JR NC,TONE1	
B01F	FEA0		CP A0h	;upper limit Tone 3
B021	300C		JR NC,TONE2	
[illegible]	110002	TONE3	LD DE,0200h	;ink color for Tone 3
[illegible]	[illegible]		[illegible] (5CB0h),DE	;spare byte = ink
B02A	21A7B0		LD HL,TABLE3	
B02D	1822		[illegible]	
B02F	110005	TONE2	LD DE,0500h	;ink color for Tone 2
[illegible]	ED53B05C		[illegible] (5CB0h),DE	
B036	21C7B0		[illegible] HL,TABLE2	
B039	[illegible]		[illegible]	
B03B	110003	TONE1	LD DE,0300h	;ink color for Tone 1
B03E	ED53B05C		[illegible] (5CB0h),DE	
B042	21E7B0		[illegible] HL,TABLE1	
B045	180A		JR ROUTE	
B047	110006	TONE0	LD DE,0600h	;ink color for Tone 0
B04A	ED53B05C		[illegible] (5CB0h),DE	
B04E	2107B1		LD HL,TABLE0	
B051	3A27B1	ROUTE	LD A,(MODE)	;route depends on mode
B054	FE01		CP 01h	
B056	2805		[illegible] Z,SKIP	;if mode 1 go to skip
[illegible]	E5		PUSH HL	;save table address
B059	CD7FB0		CALL PULSE	;display pattern

program continued on page 11...


WINKJET 1
The WIDJUP Co.
1120 Merrifield S.E.
Grand Rapids, MI

LIGHT SHOW 2000

program continued from page 9

```
B05C    E1                      POP HL          ;restore table address
B05D    CD7FB0      SKIP        CALL PULSE      ;remove pattern
B060    189E                    JR START        ;read next tone
B062    7E          SCREEN      LD A,(HL)       ;A = temp. attribute
B063    E638                    AND 38h         ;A = paper only
B065    57                      LD D,A          ;D = paper
B066    7E                      LD A,(HL)       ;A = temp. attribute
B067    E607                    AND 07h         ;A = ink only
B069    07                      RLCA            ;move ink attribute
B06A    07                      RLCA            ;   to paper position
B06B    07                      RLCA            ;     in A
B06C    BA                      CP D            ;ink = paper?
B06D    2006                    JR NZ,NSAME     ;if not make ink=paper
B06F    ED5B805C                LD DE,(5C80h)   ;DE = ink for tone 0
B073    1805                    JR LATTR
B075    3A8D5C      NSAME       LD A,(5C8Dh)    ;A = perm. attribute
B078    77                      LD (HL),A       ;set attribute file
B079    C9                      RET             ;done
B07A    E6F8        LATTR       AND F8h         ;A = all but ink
B07C    82                      ADD A,D         ;A + D (ink) = attr.
B07D    77                      LD (HL),A       ;set attribute file
B07E    C9                      RET             ;done
B07F    0610        PULSE       LD B,10h        ;# of graphics to pulse
B081    5E          PULSEA      LD E,(HL)       ;load DE with the off-
B082    23                      INC HL          ;   set of location in
B083    56                      LD D,(HL)       ;   attribute file
B084    23                      INC HL
B085    E5                      PUSH HL         ;save next table entry
B086    210058                  LD HL,5800h     ;start of attr. file
B089    19                      ADD HL,DE       ;HL = location
B08A    C5                      PUSH BC         ;save count
B08B    0601                    LD B,01h        ;pulse time - course
B08D    C5          BLOOP       PUSH BC         ;save course count
B08E    0601                    LD B,01h        ;pulse time - fine
B090    10FE        LLOOP       DJNZ LLOOP      ;pause fine times
B092    C1                      POP BC          ;retreive course count
B093    10F8                    DJNZ BLOOP      ;pause course times
B095    C1                      POP BC          ;retreive count
B096    CD62B0                  CALL SCREEN
B099    E1                      POP HL          ;retreive table entry
B09A    10E5                    DJNZ PULSEA     ;finish pattern
B09C    060A                    LD B,0Ah        ;tempo time - course
B09E    C5          BLOOP2      PUSH BC         ;save  course count
B09F    0600                    LD B,00h        ;tempo time - fine
B0A1    10FE        LLOOP2      DJNZ LLOOP2     ;pause fine times
B0A3    C1                      POP BC          ;retreive course count
B0A4    10F8                    DJNZ BLOOP2     ;pause course times
B0A6    C9                      RET             ;pattern done
B0A7 - B0C6         TABLE3                      ;contains offset point-
                                                ;   ing to addresses in
                                                ;   attribute file of
                                                ;   graphics for tone 3
B0C7 - B0E6         TABLE2                      ;same as above but for
                                                ;   tone 2
B0E7 - B106         TABLE1                      ;same as above but for
                                                ;   tone 1
B107 - B126         TABLE0                      ;same as above but for
                                                ;   tone 0
B127      01        MODE        DEFB 01h        ;variable for mode
```

NOTE1 As only the pattern for Tone 3 has 16 graphic locations, dummy locations have been provided in the other tables to bring their total up to 16. This offset + start of attribute file points to the Printer Buffer. The reason for maintaining a count of 16 for all Tones was to produce a simpler code and to provide a similar timing for displaying all tones.

FALL PRODUCT ROUND-UP

SOFTWARE

PIXEL SKETCH And GRAPHICS EDITOR Version 2.0

Reviewed by Duncan Teague

What if Michelangelo and Leonardo da Vinci had owned some good graphics software? Would they have utilized a computer for their creative efforts? Would the Sistine Chapel be covered with fanfold paper? Would the Mona Lisa be stored as electrons on a disk instead of pigment on a canvas?

Those gentlemen were the masters of their media. Unless you have a comparable skill with pallette and paint brush, maybe you'd better rely on joystick and keyboard. PIXEL SKETCH And GRAPHICS EDITOR Version 2.0 employs both.

PS/GE lets you create, edit, and label original graphics and modify, merge, and analyze existing screens with electronic tools. Some of the editing functions on PS/GE are found in Apple graphics software like Mouse Paint and Dazzle Draw. PS/GE even provides some functions the 128K Dazzle Draw doesn't include.

PS/GE operates in three somewhat compatible modes: standard color, extended color, and high resolution. Mode can be selected once the program has loaded. The standard color and extended color modes can be electronically switched at will while you're working with PS/GE. The high resolution mode must be maintained once it is selected.

In the standard color mode, one ink color and one paper color are allowed in each 8 pixel by 8 pixel character position. The brightness and flashing attributes may be on or off.

In the extended color mode, PS/GE creates eight "elements" within each character position. Each element is 1 pixel high by 8 pixels wide. One ink and one paper color and one bright and one flash attribute are permitted in each of these elements.

One screen character could thus be printed with 8 ink and 8 paper colors with alternating bright and/or flashing horizontal elements. In this mode it's possible to create new colors on the screen by juxtaposing the appropriate ink and paper colors. The manual suggests using red ink on green paper to make brown.

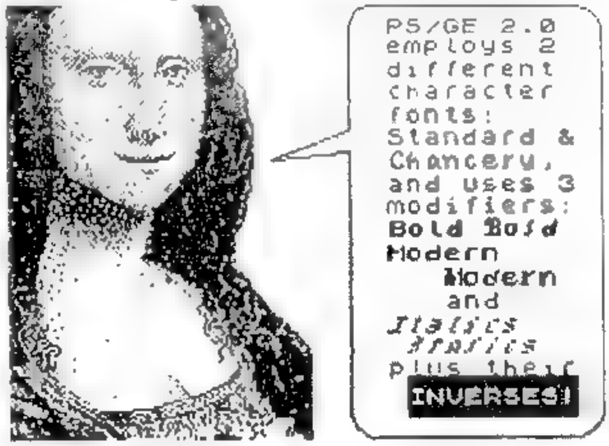
The high resolution mode amazes me. Although the hardware limits you to one ink and one paper color on the screen at any time, the software extracts the finest detail possible from the T/S 2068: 512 pixel horizontal resolution. All three modes offer the standard 176 pixel vertical resolution.

The incompatibility between modes is seen in the input and output routines. A standard mode picture is saved or laoaded as a standard SCREEN$: 6912 bytes at address 16384. An extended color graphic is saved or loaded as two files, each having 6912 bytes. One file at 16384 holds ink (pixel) and paper (no pixel) data. A second file at 24576 stores extended color and attribute data.

A high resolution display also saves or loads as two files. They extend the identical length and reside at the same starting addresses as an extended color screen. The first records the odd number column data; the second, the even number column data. The program tape includes a utility, PS/GE-32/64, which will convert one or two standard mode screens into a high resolution screen. (The listing for this utility appears in the March/April 86 TDM.)

If you've ever used MacPaint or a MacIntosh or Mouse Paint on an Apple II, you'll be right at home with PS/GE. Although the former programs employ a mouse for input instead of a joystick, they offer nearly the same drawing, text, and editing functions.

The drawing functions of PS/GE are as follows:



Mona Lisa courtesy of "Art For All Ages" by R. Conlon for Games to Learn By. I loaded the screen, mirror imaged half, block-copied, and block-erased. Then I added the text balloon.

1. Plot and Erase (free hand sketching and erasing); 2. Draw (disconnected) and Connect (-ed straight lines); 3. Circles; 4. Draw Arcs; 5. Fill/Shade (with textured patterns); 6. Paint (with solid colors); 7. Text (labeling).

The ink and paper color and the flash and bright attributes can be changed at will. Cursor speed can be adjusted from moving one pixel at at time to four times that rate. Cursor speed could be further changed by altering the program listing.

The plot (sketch) command is controlled by the fire button on the joystick. The joystick is also used to select other functions from a menu screen accessed by pressing the ENTER key. Keyboard commands can also change the cursor speed and activate the erase function without having to leave the drawing screen.

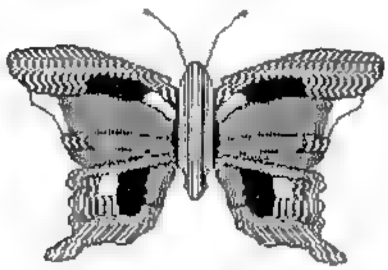
Two status lines at the bottom of the screen provide a constant readout of the cursor's position, wether the plot and/or erase functions are in use. The cursor's coordinates are important to know. Most of PS/GE's editing functions operate optimally only when the edges of drawn figures coincide with the normal character position boundaries. I'll explain later.

The text you use to label your creation can be the ordinary system font, or you may load an additional font from the program tape. The extra font is called "Chancery". It looks like it flows from a calligrapher's pen. You can alter either font from the menu. You may choose bold, modern, and italic versions of either the system font or the Chancery font at any time.

The editing functions of PS/GE are operated within an adjustable, but not elastic, window. PS/GE's window moves 8 pixels at a time. Window boundaries are always aligned with the edges of the normal screen character rows and columns. If a portion of an area to be edited extends beyond a normal character row or column, the editing window must be large enough to extend to the next row or column boundary.

Here is a list of PS/GE's window editing functions: 1. Block Copy (cut and paste); 2. Block Erase; 3. Block Rotate (90 degrees clockwise); 4. Mirror Image (horizontal only); 5. Inverse (exchange ink/paper); 6. Wide View (shrink); 7. Zoom (enlarge); 8. Digitize (analyze like a UDG-help create sprites?!); 9. Merge (a portion of one screen with another).

Mirror Image Butterfly



Drawn with PS/GE 2.0

The butterfly is my own drawing using PS/GE. I created the left side, filled and painted, and then mirror imaged it. Then I added the text.

When you select any of these functions from the menu, a window appears on the drawing screen. The window's size can be adjusted in one dimension at a time by using the unshifted arrow keys. The "S" and "L" keys will make the window smaller or larger by changing its height and width simultaneously. The joystick places the window in the appropriate position.

After using any of the editing functions, you'll have a chance to reconsider. A "SAVE?" prompt will appear, and you may "undo" the last procedure by responding with any key except "Y". The drawing functions, except Plot, Erase, and Text, can also be undone.

Hard copy can be obtained of any screen in any mode. Only the ink/paper pattern is reproduced. Colors are not represented by different dot patterns as in Tascopy or Z-Print 80. The screen can be printed on the 2040 printer or in small and large sizes on 80-column printers.

If you want to use an 80 column printer, you'll become more familiar with your printer's manual than you used to be. You'll need to know how to adjust the line feed pitch and how to send the appropriate commands for bit graphics.

My printer has to know how may bytes will follow the bit graphic command. The correct number for my C. Itoh 8510 is 256 in the standard color mode, and 512 in the high resolution mode. Those numbers had to be doubled for the large printout.

To make the large printout of a high resolution mode screen (1024 bytes per line), fit on my printer paper, I had to set the printer's DIP switches for proportional characters. This gives a print density of 1280 dots per 8-inch line, slightly more than required for this mode.

If your bit graphics mode prints each line upside down, as mine did, there's a simple solution. Alter the programs's machine code with the following POKE's, which are courtesy of program developer Stan Lemke:

Memory Address	Old Value	New Value
42919	249	193
42923	241	201
42930	249	192
62941	144	128
62966	144	128

AERCO FD-68 owners will easily be able to convert PS/GE to disk. The utility for converting standard screens into high resolution screens is another matter. PS/GE-32/64 uses OUT 255,0 and OUT 255,54 to alternate between 32 and 64 column modes. The FD-68's OUT 244,1 command interferes. Disk access must be switched off with OUT 244,0 before performing the conversion process. Loading two screens is much more difficult. I used short machine code routines to store one at 40000 and then recall it for conversion.

I really enjoyed using PS/GE. Cursor movement is slow, especially across the 512 pixel-wide high res screen, but the sophisticated editing functions surpass those of any other T/S 2068 graphics program I own. The functions for creating and editing screen segments, merging one screen with another, converting standard mode screens to high resolution screen, and printing out with excellent dot density exceed my present ability to exploit them. But I'm learning. My joystick finally has something to do besides play games.

PIXEL SKETCH And GRAPHICS EDITOR Version 2.0 is available from Lemke Software Development, 2144 White Oak, Wichita, KS 67207. The T/S 2068 program comes on a cassette with users manual for $19.95 ppd. A joystick is required.

TIMACHINE — A BASIC Compiler

Reviewed by Michael E. Carver

Deja vu! That was my first thought upon opening the large envelope from editor Tim Woods. Let's take a trip via H.G. Well's time machine by setting the controls to travel back in time one year. Exactly one year ago, I was asked to review a BASIC compiler for the T/S 2068 called ZIP (Sept/Oct '85 issue of TDM). Back to the present! I now have the task of reviewing a new BASIC compiler for both the 2068 and Spectrum (two different versions on the same tape). It's called TIMACHINE.

First, a short review. BASIC is the resident ROM language in the Sinclair machines. BASIC is a language we humans can easily use to make the computer and its processor perform a desired task, and is a fairly effortless language to learn and use. It is also a fairly forgiving language, especially with the help of Sinclair syntax and error checking. The trade-off for this simplicity is a lack of speed and flexibility. The actual resident language of the Z80A CPU (the main brain of the Sinclair machines) is machine code, also known as assembly language. This "language" is composed of about 50 different instructions, though most have many variations. The advantages of machine code include fast execution, efficient use of memory, and freedom from the dictates of the Operating System. The other side of this coin are the following disadvantages: programs are hard to understand and follow, a simple manipulation of data may involve many complicated steps, real-number calculations can be difficult and it can be very unforgiving. Programming in machine code can involve extensive study of the machine, books and tables, developing tools (assemblers and monitors), and, of course, patience.

Enter the BASIC compiler, which attempts to marry the advantages of both BASIC and machine code, while trying to avoid their drawbacks. TIMACHINE is the best compiler I have seen to date for the Sinclair Z80 machines. Timachine will compile virtually all of the Sinclair BASIC commands into a much speedier program. This compiler is quite different from others I have seen in both speed and versatility. Where many compilers only allow the use of integers (whole numbers from -32768 to 32767 or 0 to 65535), Timachine will allow the use of real numbers (decimals and numbers far larger or smaller

Now at last . . .

The FootePrint Printer Interface

The FootePrint Printer Interface was originally described in the January-March 1985 issues of SUM Magazine. Now improved and professionally built, it is available direct from the designer! FootePrint plugs into the cartridge slot of the TS-2068 and works with **both** Tasman (B and C) and Aerco print driver software. Just load the software and print. No POKES required. No modifications.

- for Centronics parallel printers
- works in both 2068 and Spectrum mode
- compatible with OS-64 & Spectrum emulators
- EPROM socket and on/off switch on board
- requires no modifications to computer
- plugs into cartridge dock—door completely closes with cable running back under computer
- frees up rear edge connector allowing other peripherals to be used; less chance of a crash
- print driver software for LPRINT, LLIST, and COPY included for 2068 and Spectrum modes

FootePrint Interface w/software & cable .... \$45.00 postpaid
FootePrint with OS-64 option included .... \$65.00 postpaid
Bare board & instructions only .... \$15.00 postpaid
Cable only for use with bare board .... \$15.00 postpaid

All prices are pre-paid and include shipping charges. Florida residents must add 5% state sales tax.

FOOTE SOFTWARE
P. O. Box 14655 — Gainesville, FL 32604
904/462-1086 (6 pm - 9 pm EDT)

DUNGEON OF YMIA

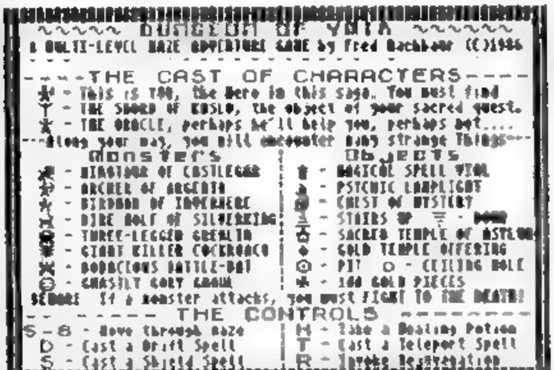
A MULTI-LEVEL MAZE ADVENTURE GAME by Fred Nachbaur (C)1986

FOR THE TIMEX TS1500

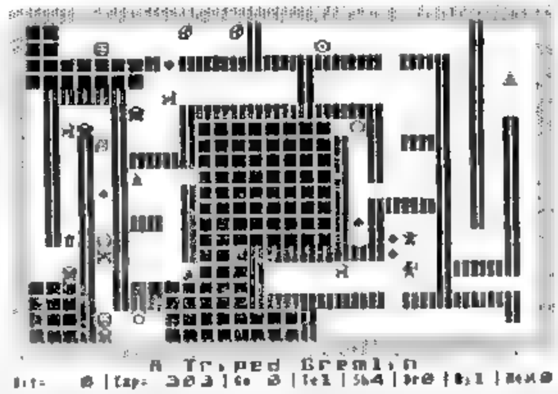
FINALLY! A FULL-FEATURE, HIGH RESOLUTION DUNGEON GAME FOR THE TS1500!

This 24K game, written entirely in machine-code, is the most spectacular program ever written for the TS1500. Nine levels, 16 types of monsters, 14 objects, six spells. Easy to play, difficult to master. Includes FAST-SAVE with auto-boot to save in-progress games; time to load entire program reduced to 70 seconds! Revolutionary TRUE HI-RES puts your TS1500 on a par with much larger machines.

Send \$24.95 (cheque or MO) to FRED NACHBAUR, C-12 MTN. STN. GROUP BOX, NELSON BC V1L 5P1 CANADA. Specify version: V1 (TS1500 + 8K Hunter NVM or equivalent) or V2 (TS1500 + 16K RAM pack). V2 requires a minor hardware addition (included). COMING SOON: V3 for ZX81/TS1000. Inquire. *** ALSO AVAILABLE: TS1500 HI*RES EXTENDED BASIC (\$16.95)



ALL ARTWORK IN THIS AD FROM ACTUAL SCREEN DUMPS!



than 16-bit numbers). There are also floating-point (or real number) compilers available for the Spectrum, but unlike Timachine, can not compile both integers and real numbers, and run only 3 to 5 times faster. Timachine allows the user to specify which numbers are to be real or integers, thus greatly speeding up performance when doing integer mathematics.

Unlike other compilers, Timachine also allows string arrays and defined functions and 2-dimensioned arrays, along with many other commonly used BASIC commands. Because it allows floating-point mathematics, one can also compile trigonometry functions (TAN, COS, ACS, SIN, ect.). In fact, according to the manual, Timachine will support all but the following BASIC commands: CLEAR, CONTINUE, ERASE, FORMAT, LIST, LLIST, LOAD, MERGE, MOVE, NEW, RESET, RUN, SAVE, VERIFY, FREE, ON ERR, and VALS. A few other commands may have some limitations connected with them. (e.g.; RESTORE, GOTO, and GOSUB must be followed with a valid line number and not an expression or variable; an array can only be dimensioned to one set length; a defined string variable may not be later dimensioned; VAL A$ is not supported.)

Speaking of the manual, I must compliment Novelsoft and Cameron Hayne (the author of the manual and program) for providing an extensive and easy-to-follow manual. The manual contains 52 pages of excellent step-by-step tutorials (sample programs included on tape), thorough explanations of commands and directives (even explaining how to obtain certain Sinclair Keywords), detailed notes on how the compiled code handles certain BASIC instructions, clear and helpful hints, definitions of Error Messages, a list of helpful POKEs, a memory map, and a list of the runtime routines.

Timachine is loaded into the memory location normally reserved for BASIC (right after the system variables) with the normal 2068/Spectrum memory map shifted upwards to allow for BASIC programs. There is approximately 27k available for a BASIC program (30k on the Spectrum). Once loaded into the computer, Timachine is completely transparent. One can LOAD or type in a BASIC program and RUN it as if Timachine was not in memory. Timachine is accessed through direct commands prefaced by an asterisk, "*" (e.g.; [*] will compile a BASIC program). In fact, the only time Timachine makes itself apparent (except for less available memory) is when the trace (an interrupt-driven program) is on. While running a BASIC program, the trace will log and type the program variables every 1/60th of a second (1/50th on the Spectrum) and provide a listing of the variables with their type (i.e.; real, integer or positive integer) and the length of string variables. This is a very helpful tool. There is a limit of 255 simple numeric variables, whose name can be any length and the standard number of string variables.

Directives to the compiler (instructions) are included in the actual BASIC program in REM statements with an exclamation point, "!", following the REM (e.g.; 10 REM ! OPEN # will start compiling at this point). Some of these directives are instructions on where to halt or re-start compiling (allowing access to BASIC or ones own machine code routines), maximum length alloted to a string variable, and setting types of numeric variables (i.e.; real, integer, ect.). One can direct a listing of the addresses for the runtime routines and the machine code variables used by the compiled program. Also, a listing of the execution addresses for individual compiled BASIC lines can be obtained. Once can specify the address at which the compiled code will reside, giving flexibility in locating ones own machine code or BASIC routines.

Learning to use Timachine is simple, but practice and study is needed if one plans to master its uses. Simple BASIC programs are easily compiled into fast-running programs. An understanding of real numbers and integers is needed to obtain maximum and exact results. When real numbers interact with integers, interesting, but usually unintended, results can occur. While testing Timachine, I used various BASIC programs I had already developed and debugged. During the first pass, Timachine checks the BASIC for any unsupported BASIC commands and provides clear Error Messages displaying the offending BASIC line, usually with a flashing "?" cursor marking the part in question. The next pass is a dry run to fix the amount of memory needed for the final version and check for destination addresses for GOTO, GOSUB, ect. commands. The last pass is the actual creation of the machine code. The user is provided with information on the length of the compiled code, amount of memory allocated for variables, length of the BASIC program, and instructions on how to SAVE, LOAD and run the compiled code. This complete compilation process is quite speedy. Timachine compiled the demo program included with the ZIP compiler in 9 seconds compared to the 31 minutes taken by ZIP (see Sept/Oct '85 TDM, pp. 18-19). I was able to compile most of my test programs satisfactorily, with only minor modifications to the BASIC. However, the one larger and complicated of the BASIC programs proved to be too convoluted to simply modify. I did not have time to fully test this program, but feel it would require a major reworking to obtain proper results via Timachine. If the program had originally been written with Timachine in mind, I see no reason it could not be easily compiled.

As the compiled code is in machine code, it can be unforgiving and provide undesired results. In BASIC, "PRINT H$ (X)" will print H$(1) if x=.5, where the compiled version will attempt to print H$(0). Another problem I encountered dealt with FOR/NEXT loops. In BASIC, one can leave a FOR/NEXT loop, jump into the middle of another FOR/NEXT loop using the same variable and upon execution of NEXT (X), resume operation at the start of the second loop. However, in machine code the continuation is at the start of the first loop. There are many runtimes (machine code routines used by the compiled version to execute selected operations) which use extensive ROM routines. This allows for simple conversion and efficient use of memory, but can slow down performance. When these runtimes are used, the improvement in speed is slight, (e.g.; CIRCLE, DRAW, COS, ect.). It is up to the user to develop a BASIC program that will utilize a more efficient compiled version. This will come with practice and experience, though knowledge of machine code will be helpful. One can use Timachine as a tutor on how to write their own machine code utilizing ROM resident routines.

Timachine is the most comprehensive, flexible compiler I have seen for the Spectrum or TS 2068. It is always a pleasure to encounter a program of this class, and I must applaud Cameron Hayne for obtaining so much from our humble Sinclairs. Depending on the programming skills of the user, one can compile fast and efficient machine code programs, though not necessarily using less memory. It is a program that will allow the novice BASIC programmer some degree of success, while allowing the more experienced programmer greater flexibility. One should not expect to produce amazingly impossible feats from this product, as these usually come from direct manipulation of the processor via ones own machine code. One will be able to produce effects that are available in BASIC, but a speeds that will greatly enhance them. In last year's review of ZIP, I stated that one should choose a compiler based on its limitations. Well, Timachine has few limitations and is a quick and comfortable program to use. Will this be the "last word" in compilers? I don't know. Let's set our "time machine" for one year into the future...

TIMACHINE is available for $19.95 + $3.00 S&H (U.S.) from Novelsoft, 106 Seventh Street, Toronto, Ontario, Canada M8V 3B4. 416/259-8682.

LARKEN TS1000 Disk Drive Interface

Reviewed by Sean W[illegible]zel

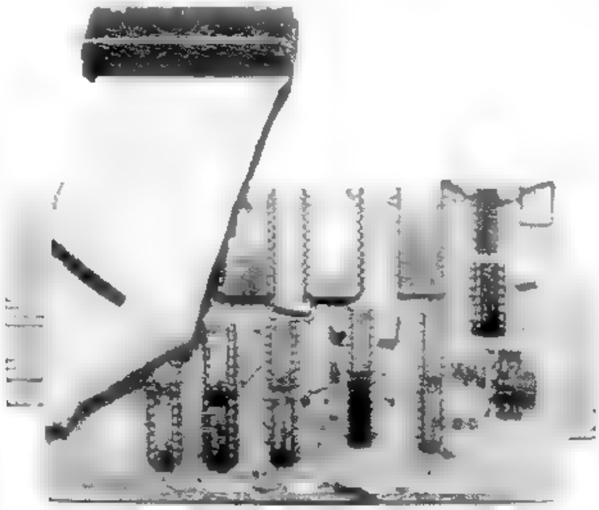
[illegible] sure all of you have thought about the speed [illegible] that [illegible] and you had to be an ol[illegible] [illegible] and get one.

[illegible] of Larken [illegible] has a real live [illegible]

[illegible] I [illegible] it at my door, I [illegible] the package wrapped [illegible] board about 3x6 inches, a power connector [illegible] the disk drive, a disk, a ribbon cable for connection to the [illegible] the drive, and a manual. I opened up the manual and found out that I now needed to buy a power supply and a [illegible] compatible disk drive. The power supply [illegible] power supply [illegible] easily obtained at almost [illegible] I [illegible] powered [illegible] there [illegible] and the drive whirred [illegible] the manual and found out that LARKEN DOS is extremely [illegible] and easily lends itself to modification by the [illegible] you a brief description of [illegible] which are actually [illegible], are provided [illegible]

[illegible] things such as [illegible] that I have made to my [illegible]. The [illegible] of [illegible] or losing that program you just spent the last 3 hours typing in [illegible] the "5 minute save").

The Larken Disk Drive Interface is available from Larken Electronics, RR#2, Navan, Ontario, Canada K4B 1H9. Phone. [illegible] 835-2680. Price is $99.00 (U.S.). Add $5 for 2-drive version (an upgrade is available for $10, if you change your mind later).



ZX-CALC + R.F.R.G.

Reviewed by D. Hutchinson

ZX-CALC + R.F.R.G. is a complex and comprehensive spread[illegible] [illegible] package.

[illegible] states that it [illegible] several [illegible] sheet package, that it does soak up memory...a lot of memory. In fact a 64K Ram Pack is a must. Also, printing is [illegible] printer. Be prepared to have a lot of rolls of the [illegible]

[illegible] has many [illegible] and features (probably more so [illegible] total [illegible] the cursor and [illegible] sophisticated spreadsheet, and at the [illegible] and handling), it presents [illegible] the most flexible spreadsheet I've [illegible] a version available for [illegible]

[illegible] got [illegible] Pennsylvania Ave., No.204, Miami Beach, FL 33139, 305/531-6464

"More About...The Mystery of the Missing 253"

by Wes Brzozowski

THE GREETING

Welcome back to another episode, as we try to unravel a few more clues about the Extended Bank Switching for the Timex Sinclair 2068. This time, we'll be getting heavily involved in how the bank switching hardware would have worked, making this installment the most complicated of the series. But this article will cover a lot of subjects, and if one item seems hazy, just skip it and move on to the next. With some rereading, things WILL get clearer, so don't get discouraged. And don't forget that the order that's easiest for YOU to learn these things, may be different from that of others. Keep rereading, and learn in your own way.

Since this kind of information hasn't been published elsewhere, I've had to invent my own notation for a lot of things. These were covered in Part 1, but if you've missed it, you can still get the back issue--July/August 1986 for $3.00 from TIME DESIGNS MAGAZINE.

This paragraph is for those who may have written or called me with information/advice/questions. If it appears that I'm ignoring you in this column, I must beg you to remain patient. Most of this second installment will have been written before Part 1 has even been put into print (publication delays, you know). As such, there's a good chance you'll have "missed" being mentioned in this installment. But rest assured that I do appreciate your interest, and WILL get to you in Part 3.

Some of you who've been looking up my page references for the TS2068 Technical Manual have probably been a bit befuddled. If you bought your manual from Timex, everything will be fine. However, the new version from TDM has the pages re-numbered a bit, and the page numbers I gave last time won't quite match up. I wasn't aware of this when I wrote Part 1, and will give the section numbers instead, from now on. I hope no one was inconvenienced by this. In order to accomodate everyone, let's define yet another notation. From here on, Technical Manual references will be abbreviated. The expression "TM3.3.2" would then refer to section 3.3.2 of the TS2068 Technical Manual.

By the way, I do hope no one is grumbling because of the renumbering trick. In doing this, our good friends at Time Designs have been able to reduce the total number of pages in the manual, and so perhaps they can avoid actually losing money on the venture.

And now, on to the good stuff!

A (NOT SO) QUICK DESCRIPTION OF THE RAM RESIDENT CODE

Let's first turn to page 255 of the User's Manual that came with your TS2068. The memory layout shows two blocks called the Utility Function Dispatcher, and the Bank Switching Code. They originally come out of the EXROM, and are copied to RAM during the computer's power-on initialization. The two memory maps on page 254 refer to these as "RAM Resident Code", and show that they may reside in two possible memory locations. To make this easier, the EXROM contains a routine that can relocate the code for us. Well, almost. The "relocator" fouls up on a couple of routines when it moves them to high memory. We'll discuss how to fix these in a future installment. Nevertheless, a short look at them now will make other things easier for us to understand.

The function dispatcher is a prime example of the right pew in the wrong church. In most computers, CALLing ROM routines directly through their memory addresses is considered about as civilized as blowing one's nose on the tablecloth. This is because later ROM versions may change the locations of the subroutines, rendering your programs unworkable. This was precisely what happened when Sinclair changed the ROM on the early ZX81s. (If you remember this, you're a true "old timer".)

The "proper" way to get at ROM routines is to pass up your CALLs through an "Operating System" that can find the routines, no matter what ROM version is in place. This wouldn't give you access to all of the ROM, however, and so requires an extra measure of programming discipline.

Is it worth it? Only when handled properly and consistently. A very similar kind of discipline allows many programs that run on a "plain vanilla" IBM PC to also run on the PC jr, and the PC-AT, which are all radically different from one another, from their disk systems, right down to their ROMs. It also allows the programs to run on the "PC Clones", that have VERY different ROMs in them. While this programming discipline means a bit more work, it has great advantages.

The TS2068 Function Dispatcher is a scaled down attempt to mimic this portion of an operating system. As mentioned last time, it's likely that at least someone at Timex hoped to rewrite the ROMs. The Function Dispatcher may have been a way to insure software compatability. By sending a "function number" to the dispatcher, the proper routine can be accessed. It also contains presently unused abilities to pass and receive data from the routines it controls. Those future ROMs may well have tapped this ability. Note that TM3.3.2 contains a reference to "the original TS2068" (as it describes OUR machines). Follow-on machines were certainly planned.

But we Timex enthusiasts, ever the unruly lot, totally ignored the Function Dispatcher, happily CALLing anywhere we liked. While the Function Dispatcher might make it easier to get at the ROM if we were running in one of the (presently nonexistent) expansion banks, it's otherwise fairly useless.

We would only use the Function Dispatcher to protect our programs against ROM address changes. But instead, no one uses it, and no one is protected. Therefore, no one will market a ROM or EPROM with address changes, because precious little software will run on it. And therefore, we needn't worry about ROM changes, and can CALL the ROM to our heart's content. It was a noble thought, Timex, but it was a bit like trying to domesticate a mongoose.

The block called the Function Dispatcher also contains some code that allows the maskable interrupt to work properly when the EXROM is switched in. It will also work with expansion banks, if they have a copy of the code at X0038 at their own location 0038. (The initialization code was supposed to copy this code into RAM expansion banks--unfortunately, it misses a byte, and anyway errantly tries to copy from the RAM bank to the EXROM; a truly useless exercise.) The interrupt code makes considerable use of the rest of the RAM Resident Code to manage the necessary bank switching.

Following this, almost as an afterthought, is a copy of the NMI handler at Home ROM location 0066. This inclusion is somewhat perplexing, as the Home ROM already has it, the EXROM doesn't link to it, it's short enough to be easily included in any expansion bank, and it doesn't work, anyway. The widely publicized NMI bug, first seen in the Spectrum and perpetuated in the TS2068 Home ROM has been faithfully copied here. There may be some subtle reason for the NMI handler to be there, but it's more likely that a Timex programmer, feeling the pressure of overdue schedules, included it without actually understanding it. At best, it reserves space for some proper code to be put later, but to us it's fourteen orphan bytes of code that are NEVER used.

Following the Function Dispatcher is the Bank Switching Code, which will be quite useful in this series. This code is a bare-bones memory manager which, with a little bit of extra flesh (and a lot of debugging), would shield us from the "hardware realities" of bank switching. While it's fairly easy to write our own machine code to switch the standard banks, the expansion banks are another thing altogether. But by always using the Bank Switching Code, we should never have been able to tell the difference. The code contains portions to do standard bank switching, portions to access the expansion hardware, and enough "smarts" to know when to do either. As such, bank switching is changed from an occasional migrane to a constant minor irritation.

Ironically, it would be better to describe the "useful

stuff" next time, when we'll be concentrating almost completely on the system software. But as a quick description, the code allows us to switch banks, move bytes between banks, find out which banks own which chunks, do the equivalent of CALL and JP functions to other banks, and other necessary niceties. Flowchart 2 (which we'll discuss next time), shows how the BANK_ENABLE routine works. This does the actual bank switching for both standard and expansion banks, and after we've seen how the hardware would probably have worked, you can check the flowchart for an example of how the hardware and software mesh together.

As has been said, this code could have resided at two different locations. Normally, it starts at location 6200, but it can be relocated to F9C0. There are several reasons for this.

If we want to add code into the RAM, there are two basic places to put it and not interfere with a BASIC program being entered. One is above RAMTOP. This is so easy to do that it's the location of choice for most T/S programmers. Yet, it's almost as easy to clear a convenient memory nook down BELOW the BASIC program in memory. The RAM Resident Code can do either.

Now, the Spectrum has no RAM Resident Code, lots of programs for the Spectrum reside above RAMTOP, and the folks at Timex made a reasonable effort to convert Spectrum programs for the TS2068. (Almost ALL programs Timex released were first sold for the Spectrum.) As such, the low memory spot is preferable, as it avoids memory conflicts. This is, in fact, where we usually find the code.

Unfortunately, the convenient low memory area is right in the middle of the space used by the second display file for the extended display modes. There are hardware reasons for this. Some of these allow both display files to reside in just two memory chips, which must be faster (and hence, more expensive) than the rest. Also, the exact location of the second display file should have allowed them to employ some little used properties of dynamic RAMs to squeeze some extra speed out of them, when reading them for display data. Therefore, when the second display file is being used, the code is moved to the less preferable (from the designer's viewpoint) location above RAMTOP.

By the way, when you're switching chunks in and out, it's always necessary to have at least one RAM chunk available, to hold the machine stack. It's needed, among other things, to make CALL and RET commands work, and they work so well that we often forget about the stack altogether. The good folks at Timex sought to help us out in this regard, by moving the stack along with the RAM Resident Code. Since this code must be available, the stack always remains available with it, and we can happily forget about it, once more. The only disadvantage is that the stack size becomes limited (they allow us 512 bytes, or 256 entries). This is normally not a problem.

The ability to have the RAM Resident Code in two different locations has another advantage. Although the TS2068 only moves code to high memory when the second display file is active, you can move it (and the stack) there yourself. If you can choose to run it in either chunk 3 or 7, you don't have to tie up one of your precious eight chunks just to keep the RAM Resident Code available to you. Simply switch back and forth to whatever chunk your own code isn't using at the moment. (Of course, you'll have to keep track of where the RAM resident code IS, in any given situation.) Also, if you should return control to the TS2068 ROM, you'd do well to put the RAM Resident Code back where the computer expects to find it.

ONWARD, INTO THE PAST

Last time, we looked at how to read and write to the bank switching registers in the extended bank switching hardware. We then saw a quick summary of what the registers did, with a promise to explain them in detail, this time.

To recap, there are four input and four output registers, which correspond to four memory-mapped I/O locations. We call the registers C0, A0, 80, and 40, and they sometimes are linked to memory locations C000, A000, 8000, and 4000, respectively.

Each expansion bank has its own register set. When we write to certain registers, every bank will "pick up" the information. In other cases, when we write to a register, the information goes only to a selected bank.

To further complicate things, only writing to some "registers" will actually cause data to be put in a conventional register. In other cases, it may only change certain bits in a register, or not go into any hardware register at all! The "Bank Switching Registers" form a motley crew of circuit functions that are as different from one another as the Marx Brothers, and are just as wild when we put them together.

Figure 1 is a block diagram of a "generic" bank switching SCLD. Note that in reality, a RAM bank SCLD would have included memory refresh and address multiplexing circuitry, for dynamic RAMs. A ROM bank SCLD would have a set of chip enable signals. But the figure does contain all of the Bank Switching Registers, and these should be common to both SCLD types. It cna then be used to show how the bank switching scheme works. It also shows how the odd bank switching philosophy selected by Timex would have allowed the SCLD chip to go into an inexpensive package with a very small number of pins.

Note that this is only a block diagram, not a complete circuit layout. Also note that it's based entirely on an analysis of what the ROM software is doing. If the designers at Timex intended additional functions not supported in the original TS2068 ROMs, we'll know nothing about them. Lastly, please note that the connection to the RESET signal is probably not what the Timex designers actually planned. It's included here to suggest that there has to be some way to "disarm" all the horizontal select registers when the computer is first turned on. Otherwise they'd start out filled with random bits, and numerous banks would all try to "take over" the same memory chunks at power-on, with some very lively results. Actually, an odd bit of code in the initialization software suggests that each bank is "unlocked" after the Horizontal Select register is disarmed through software. This suggests that the SCLD should also contain some power on "lock-up" circuitry to keep each bank out of mischief until the computer straightens it out. We'll talk about this more when we look at the software that actually uses it. (See Flowchart 3.)

As we said last time, register data is sent to the Expansion Bank SCLDs one nybble at a time, to cut down on the number of SCLD pins. This means that the SCLD has to alternately steer the nybble into the right and left half of the byte it's reconstructing. We also said that sending 02 to register C0 will reset the nybble steering logic, just in case a noise pulse may have sent a "false nybble" out, messing up the steering of later nybbles.

But if this is all we do, it won't work. If the nybbles are not being read propely, then the 02 sent in to correct the problem won't get read either. This is why we said that the C0 register must interpret the 02 command, even if the nybble synchronization is faulty. It also has to be able to interpret it if it's sent as only a SINGLE NYBBLE (just the 2), since that's how the routines READ_BS_REG and WRITE_BS_REG send it.

A "proper" implementation requires all of this, though it's a job to implement. Things get much simpler if we "bend the rules", just this once. Our little trick centers around the fact that all commands to the C0 register have "0" as their most significant nybble, only the "02" command has data line D1 set,

and this command is only sent by the READ_BS_REG and WRITE_BS_REG routines, which send it in the single nybble version, only.

And so, if we agree NEVER to send the 02 command to register C0 except in the single nybble version, the hardware will be much simpler. Any time we write to the C0 register with the D1 line set, the nybble steering logic is reset. The ROM code is completely agreeable to this trick, and so the good folks at Timex may well have had the same idea. Figure 1 is drawn to reflect this simplification. Let's walk through it now.

The lower 4 data lines come in at the top, flowing to the Nybble-To-Byte Converter. Every time the select logic detects that we're writing to a Bank Switching Register, it sends the NYBBLE CLOCK signal, allowing the Nybble-To-Byte Converter to accept the nybble. Whenever the select logic detects that we're writing to register C0 with D1 set, it sends the C0-RESET-NYBBLE signal, which resets the nybble steering logic.

The functions mentioned so far are common to every bank. This means that if you're building your own expansion banks, and are putting more than one bank on a single board, they can share this circuitry. (Just thought you'd like to know.)

The Nybble-To-Byte converter reconstructs the original byte we intended to send. Whenever the "second nybble" is written in, the select logic sends out another signal. If the nybble is written to register C0, then the signal WR-C0 is produced. When it goes to register A0, then the signal WR-A0 is sent. Similar things happen for WR-80 and WR-40. Note that these signals must be timed so as not to occur until AFTER the Nybble-To-Byte converter has a byte ready to present.

Using this scheme, when we write to register 80, our value ends up in the Bank Number Access block. This block may also be shared. This works because each bank has its own number. If we wish to change the Horizontal Select byte for a certain bank, we first write the bank number to register 80 (Bank Number Access) and then the Horizontal Select byte to register 40 (Horizontal Select). Only the Horizontal Select register for the bank we have "accessed" will be changed. The bits are high active; that is, if a bit contains a "1", then its corresponding chunk is allocated to that bank.

Registers that cannot be shared have that property because they contain information that's unique to their own bank. As such, we'll refer to them as Unique Bank Registers. Those that can be shared will be called General Bank Registers. (Bank Number Access is General; Horizontal Select is Unique.)

A bank knows it's being accessed when the number in its Bank Number Access register matches another block called the Assigned Bank #. When they're equal, the 8-Bit Comparator sends the ACCESS-THIS-BANK-1 signal, which makes it possible to write to the Horizontal Select register, or to read from any of the four read-registers in that bank. The Assigned Bank # register is set from a write to register A0, but only under a very special situation that we'll call the "setup mode". We'll discuss this in the section on the Daisy Chain. Ordinarily, writing to register A0 does something very different.

When the system is in what we'll call the "normal mode", a write to register A0 sends the "Universal Deselect Byte" to all expansion banks. This looks a bit like a Horizontal Select byte, but has important differences. Each bit represents a memory chunk, just like a Horizontal Select byte, but if a particular bit contains a zero, each Horizontal Select register will leave its corresponding bit alone. If a particular deselect bit contains a one, then if ANY Horizontal Select byte has a one in that location, it RESETS it. As such, the Universal Deselect byte tells all banks which chunks they must give up.

So, if we want to give chunk 5 to expansion bank #07, we first make sure that the Dock and EXROM banks don't have it. (The BANK_ENABLE routine would first give this chunk to the Home Bank.) Then we send the hex value 20 (bit 5 set) to register A0. Now, if any expansion bank had chunk 5, it will have relinquished it. Next, we send 07 (the bank number) to register 80 (Bank Number Access) and finally we send 20 (bit 5 set) to register 40 (Horizontal Select). We have now given chunk 5 to bank 07.

Unfortunately, in the above example, we've also wiped out whatever value was originally in the Horizontal Select register. (Actually, even the BANK_ENABLE routine acts this crudely for all but the Home Bank.) If we wished to treat at least the Expansion Banks with a bit more dignity, we could have first read its Horizontal Select register by sending 07 (the bank number) to register 80 (Bank Number Access) and then reading the register pair 80 and 40. (Remember, the READ_BS_REG routine reads PAIRS of registers.) We would then have the Horizontal Select byte as it had already been set for that bank. We could then have only changed bit 5, and any other chunk that was already selected for this bank, would remain selected.

It's also possible to read the register pair C0 and A0, for the bank number presently being accessed. While the ROM software reads this pair, it only looks at bit 2 of the resulting byte. This happens to be bit 2 of register A0, and every bank has this bit grounded. If we look at the TS2068 schematic, we see that

D2 line (and ONLY the D2 line) has a 10K pullup resistor. As such, if we put a bank number in register 80 and then try to read that bank's C0 and A0 register pair, the resulting value will have bit 2=0 if the bank exists, and bit 2=1, if not. This function is used during system initialization to find out how many banks are actually plugged into the system.

If all of this looks like a programming nightmare, that's because it is. Don't forget though, that the initialization software and the RAM Resident Code will normally handle it all for us. The only people who really need to know how to directly program the expansion banks are those who plan to build their own, and have to know how to debug them.

Since the bank switching SCLD only uses address lines A13-A15, there can only be a limited number of possible Bank Switching Registers. These are E0, C0, A0, 80, 60, 40, 20, and 00. Since only the top 3 bits are actually used, E0 would be the same as F0, or E7, for example. Each of these corresponds to a single memory chunk.

But the possibilities are even more limited than this. What we've said implies that reading a register happens when we read a memory location from its corresponding chunk, and the memory mapped I/O is enabled. But running machine code in that chunk also causes memory to be read. As such, code that can activate the memory-mapped I/O cannot run in a chunk that corresponds to any register. The only routines that ever access them are WRITE_BS_REG and READ_BS_REG, which we walked through last time. These routines are part of the Bank Switching Code, and can be located in either chunks 3 or 7, so the corresponding registers E0 and 60 must not be implemented in hardware. (Nor should the Bank Switching Code be relocated outside of chunks 3 or 7!!)

Also, it's possible that an interrupt could occur during the short time that these routines enable the memory-mapped I/O. This would cause the keyboard routine in chunk 0 to be run before returning, so register 00 cannot be implemented in hardware. This leaves register 20, which is not used, and has no apparent problem with being used. All of this is mentioned because, if you've implemented the necessary registers, it should be fairly easy to try to add more for your own use. This explanation (hopefully) shows that only register 20 is worthy of any consideration, whatsoever. But note that register 20 is comparable to memory locations 2000-3FFF. If we totally forget about using 20 as a new register, it would be possible for a ROM

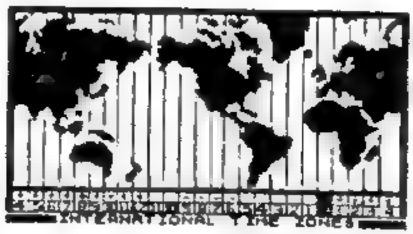
Continued on page 22...

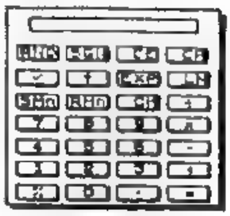
NEW PRODUCTS ANNOUNCEMENT
For TS2068 and SPECTRUM

NovelSoft announces a new dimension in capability for you 2068!

...And it's about time!

THE





WORX!

THE WORX! IS A SUITE OF PERSONAL ORGANIZATION PROGRAMS WHICH HELP YOU MAKE THE MOST EFFICIENT USE OF YOUR TIME AND THOUGHTS!

A unique combination of planning aids, decision tools, and utilities.

* Scheduler/Planner
* Notepad
* Telephone Book
* Programmable Calculator
* Decision Factoring
* Real Time Clock
* Biorhythms
* Perpetual Calendar
* International Time Zones
* Superb Manual
* Includes Spectrum & TS2068 Versions
* Supports Microdrives

A TIMELY ADDITION TO YOUR SOFTWARE LIBRARY...THE WORX!

— $19.95 U.S. Plus $3.00 S & H

NovelSoft
A FORMAT FOR THE FUTURE

106 Seventh Street, Toronto, Ontario, Canada M8V 3B4 • TEL. (416)259-8682 • CompuServe 70416,1435

bank with just a 16K EPROM to contain a completely new and upgraded version of the Bank Switching Code in those locations. (The stack would have to go elsewhere.)

At the bottom of the diagram, we see a block called Chunk Select Logic. This compares bits A13-15, which define which chunk is being accessed, and the Horizontal Select byte, which define which chunks the bank "owns". The use of IOA5 tells it wether we're really accessing memory or just a bank switching register. If the TS2068 is accessing one of this bank's chunks, then the ENABLE signal is sent out.

Note that this logic doesn't check $\overline{MREQB}$. If the TS2068 isn't accessing memory, then the ENABLE signal may switch back and forth, but it will do so harmlessly, since the memory select logic further downstream will sort it out. However, the address lines settle out a full clock cycle before the $\overline{MREQB}$ line does, and so this buys us extra switching speed. This is needed because ENABLE is used directly to generate the $\overline{BE}$ signal, and this HAS to be applied fairly early on, but again is harmless if memory isn't being accessed. (Those of us who've used the $\overline{BE}$ line in our own projects learned this the hard way; it just seemed polite to pass it on to save anyone else the trouble.)

The ENABLE signal should be sent out if IOA5 is high and A13-15 match the appropriate bit in the Horizontal Select Register. It also could optionally be sento out if IOA5 is low, A13-15 match the Horizontal Select, and the chunk in question is 3 or 7. (This would let the READ BS REG and WRITE BS REG routines run in an Expansion Bank without getting cut off in mid-instruction when they switch IOA5. No, I don't know why you'd want to do this, but you may have some good ideas that I don't.)

Figure 2 shows an entire expansion bank, including the SCLD we've just discussed. The $\overline{BE}$ signal is generated from the ENABLE line as an OPEN COLLECTOR signal, so that many banks can share the output. An alternate method in use in some products today to simulate a Spectrum Bus generates BE with a logic inversion and a blocking diode. This is also quite acceptable.

The Memory Decoding Logic will then decode the bank's memory as normal, except for one, or possibly two, additional constraints. For the first, memory is only enabled if ENABLE is active. The second possible constraint is based on educated speculation, but is still, admittedly, a bit of guesswork.

We know that the TS2068 is basically an enhanced Spectrum. Whenever possible, Sinclair's design was used, and Timex did announce that it would release its own version of the Sinclair Microdrives. This device uses its own crude version of bank

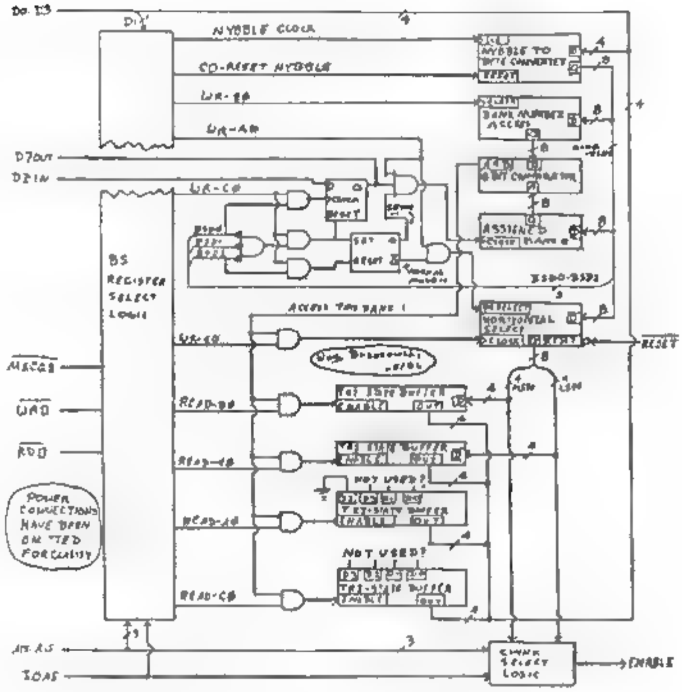


FIGURE 1. EQUIVALENT SCLD FOR EXPANSION BANKS
This suggests one way (but necessarily the only way) to implement the Bank Switching Registers

switching, werein it disables the Spectrum ROM and switches in its own when the code in the Error Handler (location 0008) is run.

The extended TS2068 commands, like LOAD *, SAVE *, FORMAT, MOVE, and CAT are implemented in the ROM almost exactly like they're implemented in the Spectrum. That is, if you know the command format, you can type them into a line of BASIC, and the TS2068 will accept them. However, they're set so that when you try to RUN them, the error handler at location 0008 will be executed. The only way to make the commands work is to switch in another ROM when the instruction is run at 0008. It must then check the cause of the "error", and run an extended command, if one is pending.

There are two ways to do this with Extended Bank Switching. We could define another special bank number (perhaps FD) which switches into chunk 0 when the instruction at location 0008 is executed. But every other expansion bank would have to contain the circuitry to check this, and switch themselves in and out, adding cost and complexity. Alternately, we could put the checking and switching circuitry only inside the microdrive interface, and give it a way to disable all banks when it switches in its un-numbered "Superbank".

The superbank method needs a signal that does to the expansion banks what $\overline{BE}$ does to the Standard Banks. The TS2068 has 3 backplane signals that are named but not wired into the computer. These are DZIN, DZOUT, and BUSISO. We'll see in a minute that DZIN and DZOUT are needed elsewhere, so let's speculate that BUSISO would have disabled the Expansion Banks. (I've heard mention that BUSISO was instead intended to tri-state U15 in the TS2068, but the schematic says it isn't wired to that chip. For the moment, let's consider this is an unreliable rumor, but I'd welcome any evidence to the contrary.)

Getting back to our memory decoding discussion, we may then guess that no memory would be enabled if BUSISO were active. The diagram shows a "Special Buffer" at the BUSISO line, because the lack of a "bar" over its name suggests that it's high-true. This means that the buffer must "see" a low signal if no microdrive interface were plugged in, leaving it floating. This is opposite to what a TTL buffer would do, although some DTL structures would fit the bill nicely. Note that if the microdrive interface were part of the BEU, then BUSISO would never be floating and the special buffer would be unnecessary.

"WHAT DO WE DO NOW, BATMAN?"

Now, all of this may be very nice, but there's still one glaring problem. When we want to send information to a Unique Bank Register, we must first put its number in the Bank Number Access register. If this matches a bank's Assigned Bank #, we can then access that bank's Unique Registers. But the Assigned Bank # is itself unique, so how do we get a value in there, in

the first place? When we first turn the machine on, that register will be full of garbage. How do we find out what it is? Worse yet, what if TWO banks "power up" with the same Assigned Bank #?

It would seem we've painted ourselves into a corner.

DAISY, DAISY, GIVE ME YOUR ANSWER DO

To our rescue comes an incredibly oddball kludge called the Daisychain. The main purpose of this whackiness is to let us put a value into the Assigned Bank # register for each bank. Since we can't use the Assigned Bank # register to access the bank at this time, each bank contains a flip flop that's one bit of a shift register (the Daisychain). Ordinarily, each bank's flip flop contains a "0", but a single "1" bit is stepped through, from bank to bank. If a bank has the "1", then we can put a value into its Assigned Bank # register.

Figure 3 shows the BEU functions that are needed to add Expansion Bank capability. It will drop the BE line if BUSISO is active, or if IOA5 is low and A13-15 indicate that the chunk being used is not 0, 3 or 7. This will prevent the memory in the standard banks from trying to "answer" an attempt to read a Bank Switching Register. The rest of figure 3 is the start of the Daisychain.

The BEU contains its own form of the C0 register. It normally operates in what we'll unimaginatively call the Normal Mode. Everything we've described so far assumes this mode. However, if we send 00 to register C0, we reset all the bits in the Daisychain and enter what we'll call the Setup Mode. This switches flip flops in the BEU and all the expansion banks. Also, DZOUT at the BEU goes high.

But DZOUT at each expansion bank is still low! Figure 4 shows how this can be. Unlike all other backplane signals, which are shared on a common bus, DZIN and DZOUT are not. This is necessary in order to retain the structure of a shift register. Unfortunately, this is not readily compatable with the normally used method of stacking additional items onto the backplane, which would short all the DZINs together and DZOUTs together, and wouldn't match one DZOUT with the next DZIN. In fact, it would seem that the most convenient method would use expansion banks on edge-connected cards, plugged into a motherboard, filled with female edge connectors.

By sending an 01 to register C0, we clock each flip flop in the daisychain, and the "1" bit moves into the next bank. When we're in the setup mode (and ONLY then) we can write the Assigned Bank # to register A0, and it will be put in the Assigned Bank # register of the bank that has the "1" in its flip flop. In this way, we individually access each Assigned Bank # register. When we're done assigning numbers, we send 04 to the C0 register, which clears all flip flops and puts us back into the normal mode.

AN INTRODUCTION TO THE SYSCON TABLE

If you haven't yet done so, read TM3.3.2, which gives a snail's eye view of the subject. The "proposed expansion banks" are the very same banks we've been talking about. The SYSCON table is a list and description of all the extra "memory" plugged into the TS2068. The LROS and AROS parts describe what you've got plugged into the Dock bank, and comprise 12 bytes. Note that each expansion bank takes up twice as many bytes, suggesting that the good folks at Timex planned to put a lot more "horsepower" into those guys.

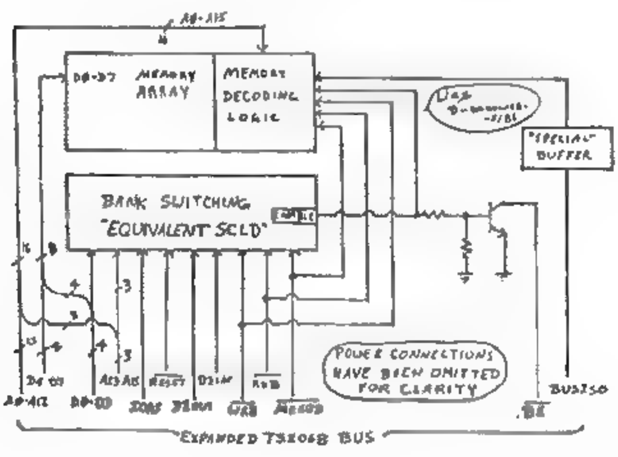


FIG. 2' EXPANSION BANK (SLIGHTLY MODIFIED?)

One thing may appear just a bit distressing. The table description says there's room for only ELEVEN expansion bank entries. Well, it's even worse than this, because the space for eleventh entry is used as a scratchpad by the initialization software. (Possibly a bug.) But if we really want more, we should note that the system variable SYSCON contains the address of this table, and we can change this, and put a larger table anywhere we'd like. Each expansion bank has a chance to run some of its own code during initialization, and one of these can rewrite the table. But the hardware that contains this bank should also contain some fancy buffering circuitry for the additional banks, or there'll be TTL fanout problems, not to mention unacceptable capacitance on the bus lines. (Actually, if you try to figure out just how many TTL chips will be needed to replace one bank switching SCLD, you may find it unlikely that even ten expansion banks will ever be run together at the same time.)

The table contains numerous options, and is laid out as follows:

SYSCON Table Configuration

2 bytes for the Dock Bank; 8 for AROS followed by 4 for LROS. See TM3.3.2)

Expansion Bank descriptions follow. One 24 byte block for each bank

01=ROM Bank ; 02=RAM Bank ; 00=Bank Inactive

1 Bank #. MSB is set if bank is not yet renumbered

The following is copied from locs 0000-0013 of ROM Expansion Banks:

2) In RAM Banks: Chunks Available, A1 true. For ROM will have bit 5 reset
3) For ROM Banks, these 3 bytes may contain a JP
4) instruction for RESET ? But not used in TS2068 ROMs!

5 Address of the Close Channel routine. It is called with PRM DU=2,
6) and Stream Number on the stack

7) Address of initialization code (perhaps to open a channel attached
8) to this bank, or to mark the bank inactive.)

9 Not used ??

0A-0C A JP instruction to an error handler? Not used in ROM

0D-0F Not used ??

10) For ROM banks: should have been the boot up address. Instead, because
of a bug, the address of this table entry is the boot up address.

11 For ROM banks Address of the power-on initialization code
(But may not reside in chunk 3)

13 Not used ??

For ROM banks: 00=Don't initialize ; 01=Initialize

For ROM Banks: Boot up priority. Low number=high priority. Home Bank=80

Interrupt Priority. Ram banks get 255. ROM get lower=higher priority

More of these 24 byte blocks, as needed

A single byte marks the end of the table:
80=End of Table)

The portions marked as not used may have been reserved for future expansion, but at least one byte was probably set aside to identify the actual function of each ROM bank. This would allow us to find, at a glance, what additional functions were actually "squirreled away" in the extra banks.

The left hand column contains the SYSCON Entry numbers. For example, SYSCON 01 contains the bank #, and every bank has its own SYSCON 01. As such, the SYSCON Entry number is not a displacement into the SYSCON table, but the displacement into the entry for a particular expansion bank. Only some of the table entries are self explanatory. Each will be discussed as we wade through its use in the ROM code.

A LOOK (FINALLY) AT SOME ACTUAL CODE

From here to the end of the series, you'll have the chance to double-check everything I've told you so far. All of my pictures, tables, and descriptions will have to be consistent with the Timex code. It's fully possible that I've missed something in my search through the ROMs, and I'll be counting on you to let me know if you see anything that looks "suspicious". Together, we can add whatever finishing touches are needed for a full description of the Extended Bank Switching.

Don't forget my promise last time that the software is fairly civilized, though somewhat amusing. If reading the hardware description has been as draining for you as writing it has been for me, we can take heart in the fact that it's all downhill, from here on!

Flowchart 1, given last time, is part of the very top level initialization code the machine runs when we turn it on. Part of the Home Bank RAM has already been intialized, and some system variables reflect this, but the memory map on page 255 of your TS2068 USER'S Manual shows "Machine Code Variables". The size of this is determined by the contents of the Dock Bank. (See TM5.1.2, TM5.1.2.2 for more information) and the system hasn't yet found out how much memory to set aside. Therefore, this, and the memory following it have not yet been set up. At this point, we check for extra memory plugged into the system:

At X0BE7 we set the initial location of the SYSCON Table. This has space for AROS, LROS, 10 Expansion Banks, and an 11th Expansion Bank area, which (possibly due to a bug) is used as a scratchpad. Its size is fixed, and if we need a larger table, we must move it somewhere else, ourselves. We then CALL X09F4 which actually builds the table (we'll flowchart this next time).

We then check the SYSCON Table for an LROS. If there is one, there are no machine code variables, so we finish setting up the system variables, and run the LROS according to its instructions (see TM5.1 for more information.)

If there is no LROS, we end up at X090F, checking for an AROS. If we find one, we check its type (see TM5.1.2.). A BASIC AROS uses no Machine Code Variables, so we finish setting up system variables, and return to Home ROM, after setting a flag telling it to run a BASIC program out of the Dock bank. A m.c. AROS uses Machine Code Variables, which we insert and then finish initializing the system variables. We then run the AROS as required.

If there is neither AROS nor LROS present, we end up at X0918, where we can initialize the system variables. At X099A, we set up so that the main execution loop in Home ROM will run after initializing (an Expansion Bank can override this, if set up properly). We then point to SYSCON 00 for the first expansion bank, and enter a loop to check each bank.

In this loop, starting at X09AC, whe check SYSCON 00. A value of 80 marks the end of the table, causing us to end the loop. If it's not 80, then we check if SYSCON 00 has the value 00. This marks the bank as inactive, causing us to point to the next bank in the SYSCON table, and loop to X09AC.

If the bank is active, we get its number from SYSCON 01. Then from SYSCON 15, we get the Initialization Flag. If this flag is 01, then we will have already run some code in that bank when the SYSCON table was built (more on this next time) and this bank may also "take over" the system after we're done initializing. This depends on its "Boot Up Priority", which we will discuss in a moment. If the flag is not 01, then we point to the next bank in the SYSCON table, and loop again to X09AC.

However, assuming that the Initialization Flag was 01, we end up at X09C4, which gets SYSCON 16: the Boot Up Priority. (The lower the value, the higher the priority.) If this is the highest priority found so far, then we save it and continue. Otherwise we loop back to X09AC.

If it IS a higher priority, we get SYSCON 10. (Note that in my flowchart I accidentally reversed the digits and called this entry 01. SORRY ABOUT THAT!!!) If the code were written properly the contents of SYSCON 10 would be the boot up address. (Where we'd run after initializing.) Unfortunately, due to a bug in the ROM, the address of SYSCON 10 is used instead. (This is a very nasty bug, but at least I can blame THIS error on someone else.) The new boot up address is saved, and we loop again to X09AC.

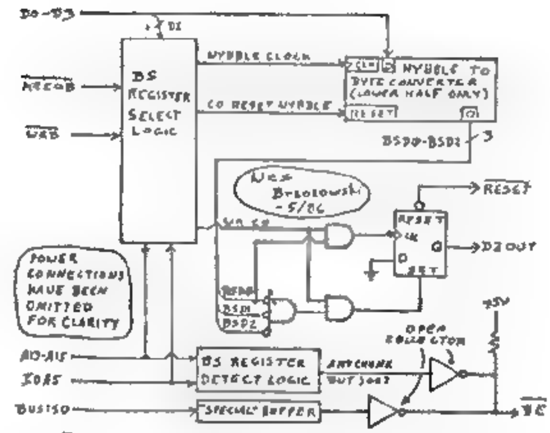


FIG. 3. BEU FUNCTIONS TO ADD BUSISO & DZOUT

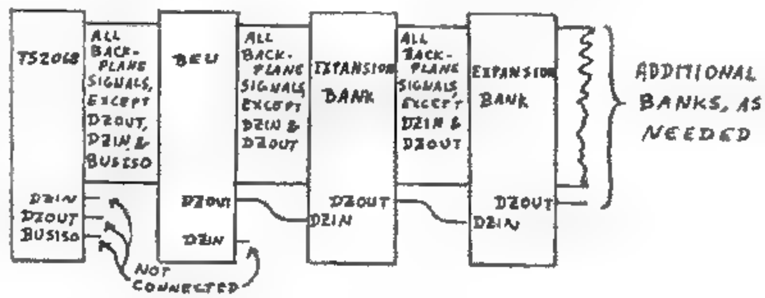


FIG. 4. TS2068 WITH EXPANDED BUS AND DAISY CHAIN

When we find a value of 80 at SYSCON 00, then we've reached the end of the table. We leave the loop, find the highest priority bank and boot up to the given address. (Default is Home Bank, at 0E2F; the Main Execution loop.)

That's the entire flowchart. I should point out one tiny "buglet" that also crept in. The box marked X09E9 should say "...Enabling 0,1,2,4,5, and 6 would...". I left out chunk 5 in a transcription error as I copied over my notes. This shows once again that it was more than just my penmanship that began to fail near the end of that long flowchart! (Is my face ever red!)

THE HOMEWORK

If you want some extra things to do, there's plenty. Walk through Flowchart 2 and use its information to continue your own annotated disassembly of the bank switching code. Try to follow what it's doing with the Bank Switching Registers (it's a fairly simple example). If you can do that, then do your own disassembly and flowcharting of the GET_STATUS routine at 6405 hex. Don't disassemble it until after including corrections shown in TM6.5.2. The Expansion Bank portion doesn't change, but the rest is a real mess, and you won't get a feel for how the routine sorts out different banks unless you include the corrections.

Read through the listings of the RAM Resident Code in Appendix A of the Technical Manual, if you haven't yet done so, and also read TM4.1 in I/O channels (yes, streams and channels figure into this subject, too).

Once again, feel free to write with questions or comments, and please include a SASE, if you wish a reply. I am Wes Brzozowski, 337 Janice St., Endicott, NY 13760. I also like phone calls, 607/785-7007...provided you don't call collect, and call before 9:30 PM, EASTERN time. See you next time!

3-D TIC-TAC-TOE

by Warren Fricke

THREE-DIMENSIONAL TIC-TAC-TOE is a variation of a very familiar game, but this arrangement has a different twist. In this version the program constructs three identical playing planes, numbered 1, 2, and 3, reading from left to right. Refer to the screen dump of Figure 1. By considering all three of these planes jointly, a player can win--or score points--when three of his marks are arrayed in a straight line, in any direction.

As in the conventional game, players take alternating turns to plot their marks, an "X" or an "O", in any of the three planes. He (or she) does so by first touching a number key designating the plane, and then a letter from the group of keys in the lower-left corner of the keyboard, the keys QWE ASD ZXC. These nine keys correspond to the same nine positions in each plane. The player does not need to press the ENTER key for a selection to be received and recognized by the computer. Touching the ENTER key is reserved for the signal to the computer to clear the screen and start over on a new game. So avoid ENTER unless a new game is what you have in mind.

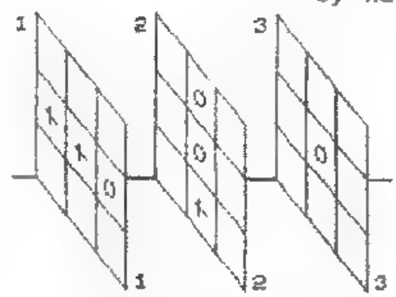
To be fair with the players, the program is designed to determine randomly, for each new game, whether the player on the left or the one on the right starts. But the starting player is not permitted to place his initial mark in the desirable center spot of the middle plane. And neither player is permitted to place their marker over one belonging to his opponent. The penalty for any of these illegal moves is forfeiture of that turn to play.

This routine contains several error traps which prevent the players from selecting an illegal number or letter. These traps are contained in lines 425, 445, and 590.

System address 23559, rather than INKEY$, was used to indicate which key was selected by the player. This approach simplifies the construction of the program, which requires a wide range of input values.

In a program of this type, the computer must make many time-consuming decisions in the principal loop that lays between lines 60 and 600. Some speed-up would ensue if lines like 500, 510, ect., contained an additional statement: GO TO 600. Such a statement would obviate the need to test any of the conditions that follow. But, the slight additional speed was not considered to be worth the effort here.

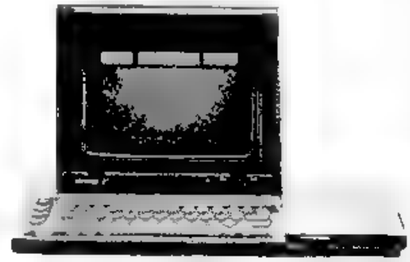
Have fun. Feel free to embellish the program further, if you so desire.



```
   2 REM ** THREE-DIMENSIONAL
             TIC-TAC-TOE
                 for
          SPECTRUM or TS 2068
             A version  by
             Warren Fricke
   3 REM **  "A-22"

   5 BORDER 1: PAPER 6  CLS
  10 FOR j=USR CHR$ 144 TO USR C
HR$ 144+15: READ a: POKE j,a. NE
XT j
  30 GO SUB 1000
  40 GO SUB 2000
  50 RANDOMIZE   LET p=1-(1 AND
RND<.5): LET t=0
  60 IF p=1 THEN PRINT AT 0,0;"R
IGH";AT 0,30; INK 2;"O": LET m$=
CHR$ 144  LET i=2. BEEP .08,0
  70 IF p=0 THEN PRINT AT 0,0;"
LEF" AT 0,30; INK 1;"+"· LET m$=
CHR$ 145: LET i=1: BEEP .08,12
 400 REM **PRINT PLAYER'S PIECES
 410 PAUSE 0
 420 LET a=PEEK 23559: IF a=13 T
HEN CLS : GO TO 40
 425 IF a<49 OR a>51 THEN GO TO
410
 430 PAUSE 0
 440 LET b=PEEK 23559  IF b=13 T
HEN CLS · GO TO 40
 445 IF t=0 AND a=50 AND b=115 T
HEN GO TO 600
 450 LET c=8*(a-49)
 500 IF b=97 THEN PRINT AT 7,5+c
  INK i;m$ AND SCREEN$ (7,5+c)=C
HR$ 32
 510 IF b=99 THEN PRINT AT 14,9+
c; INK i;m$ AND SCREEN$ (14,9+c)
=CHR$ 32
 520 IF b=100 THEN PRINT AT 11,9
+c; INK i,m$ AND SCREEN$ (11,9+c
)=CHR$ 32
 530 IF b=101 THEN PRINT AT 8,9+
c; INK i;m$ AND SCREEN$ (8.9+c)=
CHR$ 32
 540 IF b=113 THEN PRINT AT 4,5+
c; INK i,m$ AND SCREEN$ (4,5+c)=
CHR$ 32
 550 IF b=115 THEN PRINT AT 9,7+
c, INK i,m$ AND SCREEN$ (9.7+c)=
CHR$ 32
 560 IF b=119 THEN PRINT AT 6,7+
c, INK i,m$ AND SCREEN$ (6,7+c)=
CHR$ 32
 570 IF b=120 THEN PRINT AT 12,7
+c, INK i,m$ AND SCREEN$ (12,7+c
)=CHR$ 32
 580 IF b=122 THEN PRINT AT 10,5
+c, INK i;m$ AND SCREEN$ (10,5+c
)=CHR$ 32
 590 IF b<97 OR b=98 OR b=114 OR
 b=121 OR (b<119 AND b>115) OR (
b<113 AND b>101) THEN GO TO 430
 600 LET p=NOT p: LET t=1: GO TO
 60

1000 REM **TITLE & INSTRUCTIONS
1010 PRINT AT 2,7;"THREE-DIMENSI
ONAL";AT 4,9;"TIC-TAC-TOE"
1020 PRINT AT 7,3;"This version
does not keep   score nor determ
ine a winner.   Players must do
this themselves,as the rules may
 vary."
1030 PRINT AT 11,3;"The computer
 determines ran- domly whether L
EFT or RIGHT        starts the game
."
1040 PRINT AT 14,3;"Computer wil
l keep tract of  whose turn it i
s. Players alter-nate."
1045 PRINT AT 17,3;"Use the ENTE
R key only to     start a new gam
e."
1050 PRINT AT 20,3;"Stand by."
1100 PLOT 56,150: DRAW 136,0
1110 PLOT 72,134. DRAW 88,0
1120 PAUSE 800: CLS
1130 RETURN

2000 REM ** 3-D BOARD
2010 FOR m=36 TO 164 STEP 64
2020 FOR n=160 TO 88 STEP -24
2030 PLOT m,n· DRAW 48,-48
2070 NEXT n: NEXT m
2200 LET m=124
2210 FOR n=36 TO 212 STEP 16
2215 PLOT n,m-n: DRAW 0,72
2220 IF n=84 THEN LET m=188
2230 IF n=148 THEN LET m=252
2240 NEXT n
2250 FOR n=20 TO 212 STEP 64
2260 PLOT n,88: DRAW 16,0
2270 NEXT n
2290 PRINT AT 0,4;"T-HAND PLAYER
'S TURN. USE"
2300 PRINT AT 2,3;1;AT 2,11;2;AT
 2,19;3,AT 16,11;1;AT 16,19;2;AT
 16,27,3
2310 PRINT AT 18,0;"Just touch t
wo keys· 1,2, or 3 &  one letter
 key. Use the ENTER key only to
start a new game. Gowhen ready."
2350 RETURN

3000 DATA 48,76,68,66,66,34,50,1
2,144,80,40,16,24,20,16,17
```

QL KILL

```
90 gun
100 REMark PROGRAM BY RON S. MORR [illegible]
110 REMark CHICAGO DATA CENTER MAIN FRAME [illegible]
120 REMark CONVERTED TO SuperBASIC BY DAVID JOHNSON [illegible]
130 REMark WITH KIND PERMISSION FROM THE AUTHOR FOR THE [illegible]nclair QL
135 WINDOW 512,256,0,0:CLS
140 DEFine PROCedure BANG
145    CURSOR 50,50,10,10
150    CSIZE 3,1:PRINT "BANG!"
151    death
155 END DEFine
170 DEFine PROCedure gun
172    LET chambers=RND (1 TO 6)
175    LET n=0:click=RND(1 TO 6)
180    IF click=chambers THEN BANG
1[illegible]    IF click<>chambers THEN liv[illegible]
185 END DEFine
190 DEFine PROCedure live
195    CURSOR 50,30,10,10 [illegible]
200    [illegible]
210    PAUSE 100 [illegible]
2[illegible] END DEFine
2[illegible] DEFine PROCedure death
2[illegible]    [illegible]
240    [illegible]
245    PRINT [illegible]
2[illegible]    INPUT [illegible]
255    IF [illegible]
2[illegible] END DEFine
```



QL KILL (not recommended for weak stomachs) is a simulation of the ever so popular (?) RUSSIAN ROULETTE game. The listing was converted from a PASCAL program and contributed by David Johnson, with permission from the author, Ron S. Morr. David would like to get in touch with other QL users. Write to: 2399 St. Rt. 95, Edison, OH 43320.

QL Peintre

A Review
by
Paul Bingham

French language suffers the fate of English at times; being difficult to pronounce properly when read from text. So it is with QL "Pine tree", "Pain tree", "Pee-in-tray", or "Pay entry" as this program may have been so falsely introduced to you. If we could all read French it would be instantly obvious that QL "Painter" is a French screen artist program.

QL PEINTRE is a classy program in many ways. After loading begins a picture of two chimps appears: seemingly a digitized photo image. Nearly three more minutes of on-and-off file loading finally brings a blank subscreen with sharp surrounding icons. French and English titles toggle back and forth with a key press. The look is sophisticated, elegant. The icons are easily readable and easily used.

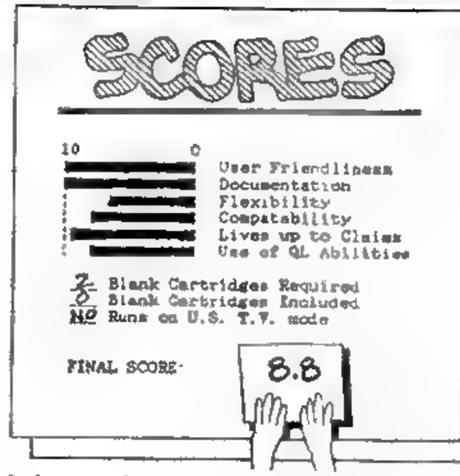
Should one require additional help, the documentation is excellent. Now this is not a flashy instruction booklet, but it is the first one I have seen which states simply what to press next and then accurately tells what the outcome will be. Even though the instructions do not mention it, I have found it helpful to have a formatted cartridge on hand for saving a sceen prior to running QL Peintre. There is a Microdrive icon, but like so many other QL software titles, it does not provide for formatting of a cartridge. It does provide a nice scrolling directory option, though!

Like GRAPHIQL, reviewed by Vince Lyon in the March/April 86 issue, QL Peintre has many, many abilities. Some are far more versatile as well. Circle and Arc drawing are so much improved! Fabulous, too, are the two type faces (one very MacIntosh) in two sizes with four spacings each. Line and spray widths with intensities are very adjustable as are the colors and textures. But keep in mind that QL Peintre is capable of only four colors in High Res mode F1, and will not run in mode F2 on an American TV. QL Peintre supports all the colors in F2 mode.

Is Peintre perfect? It does have a limitation or two. It incorporates the standard security checks requiring a backup and original present. The screen work area is much smaller than other such programs. It only provides 63% of the screen that GraphiQL uses. It also does not have a wrap-around feature when drawing. If one bumps into the work space edge diagonally, the cursor continues straight along that edge until stopped. Jotting and spraying do not quite reach the edge of this work area either. These are really minor items, however. Unlike GraphiQL, Peintre never does "growl" back!

QL Peintre isn't intended as a drafting design or CAD program. QL WORLD MAGAZINE (from England), carried a review in the May '86 issue, of six such programs. Most CAD packages were more expensive than Peintre, too. Missing from their review was a new release, TECHNIQL (from Talent...writers of GraphiQL). Such would be good future Quick Look fare.

QL Peintre has a good set of save and printer options. The Printdump program is unprotected and may be included by the user in other programs. I was amazed to find that it will load almost any saved screen, including ones from COSMOS (reviewed last issue) and GraphiQL, too. Screens can then be altered and printed out on an Epson printer. But the output process takes 22 minutes! GraphiQL's beats it by three or four minutes. But unlike other dumps, this produces an output sideways! So if your printer is like my little Epson Homewriter 10, most graphics look slightly elongated. Now with Peintre they just come out taller.

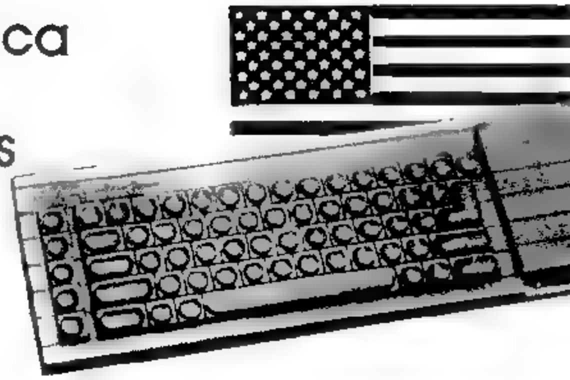


QL Peintre is sharp and professional. It has limitations but is very polished and easy to adapt to. If the programmers in France keep this up, I may start learning to read French.

The Future of the QL in America And Some QL Graphics Systems

by

Mike de Sosa



THE QL IN AMERICA

Many of us in recent years have, in anguish, witnessed the demise of first the Timex Computer Corp., then Sinclair Research USA, and finally Sinclair Research Ltd., itself. I say "in anguish" because with their failure went fond hopes of continuing professional support for our computers and dreams of a proliferation of third-party hardware and software. These organizations, blessed with an excellent product and the good will of many thousands of cult followers, just plain "blew it" and, in the event, passed up the opportunity for millions in profits. None seemed to learn from its predecessors. In each case, arrogance and poor customer relations prevailed. Statements like "we don't really need the U.S. market" and "each Sinclair employee produces millions in earnings" added to the insult of telephone calls that were not returned and letters unanswered. Many recommendations from periodicals, users' groups, and loyal customers, some based on time consuming research, were spurned. If it was not the company's idea, it was not, needless to say, a good idea. The outcome, the result of arrogance and delays due to poor management, was probably inevitable.

QL users and would-be users in the U.S. are now dependent on a single distributor offering a limited number of QLs through a dozen dealers--without factory or engineering back-up. Before taking the plunge, would-be QL buyers should be aware of a few things and then satisfy themselves that their QL will be adequately supported. Item--the QL will be replaced in a year or so (perhaps this fall in the UK) by one or more follow-on "QL-compatible" systems that do not use Microdrives. Item--when the present small stock of QLs are gone, no more may be manufactured; this will affect the availability of replacement parts and maintenance and the quantity and cost of future software for the QL. Item--there is a difference in QDOS addressing that causes many software programs to work improperly on U.S. machines: for example, QL Project Planner, QL Decision Maker, GraphiQL, and VROOM! (This problem also affects U.S. software designers trying to get their programs to work on European QLs.) Item--the Psion software programs sold with QLs in the United States are now several versions old and the documentation for these programs is older yet. Item--the QL is poorly documented in the QL User Guide, and scores of books on the QL--all written early on before the operating system was perfected and before peripherals were available--do little to help the situation. (This problem--a major shortcoming of the QL since its launch, and before it the T/S 2068--was ignored by Timex and Sinclair organizations despite urgent pleas and recommendations by QL users.) Item--many QL users are now getting bad advice...better methods of communicating authoritative information regarding the QL are needed.

I do not believe that the QL distributors and dealers are moving nearly fast enough to correct these problems, all of which are capable of easy and inexpensive solution. For this reason, prospective QL and QL software buyers should pressure the distributors and retailers to do something about all of these problems ASAP. In doing so, you would do yourself and them a favor, possibly saving them from a rather predictable self-destruction. Lets have some cards, letters, and calls on this, folks!

The following things should be done soon (and that doesn't mean "next year"):

a. Supply QLs with the latest version of Psion QL software. (It should cost very little to do this, and buyers would gladly pay an extra cost.)

b. Supply QLs with additional documentation to supplement the inadequate, frequently erroneous, and badly out-of-date QL User Guide. (Making this supplement available to QL owners here and abroad could make this a profitable endeavor.)

c. Devise a universal "patch" which would permit all European QL software to work properly on U.S. QLs. (This is long overdue--a result of inaction.)

d. Insure they are getting good technical advice regarding which QL peripherals and software to market in this country.

e. Decide whether the current U.S. version of the QL, Microdrives and all, should be manufactured further, perhaps with additional built-in or plug-in RAM and RAM-disk software. (An enhanced QL could be profitably marketed in this country for another two to three years.)

f. Make on-the-shelf QLs available with distributor installed disk interfaces, RAM cards, and RAMdisk software.

g. Select and standardize use of a disk interface with the QL and a single DS-DD drive, preferably a 5 1/4" drive.

h. Establish a toll-free "QL HOT LINE" for three or four hours a day to answer consumer questions.

i. Advertise the improved product. If nothing is done, that is, if things proceed at the present pace, my prognosis for the QL in the U.S. is a slow death.

QL GRAPHICS/CAD SYSTEMS

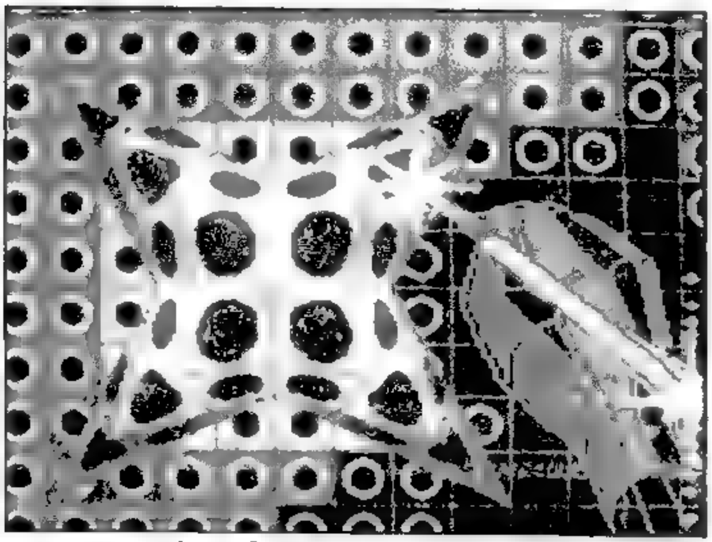
There are now many types of graphics systems for the QL, ranging from those used to draw pretty pictures to two-dimensional computer-aided design (CAD) programs to three-dimensional graphics design programs. With one exception, I believe the best of these QL graphics programs are discussed below. (The exception, QL Peintre from France, is discussed by Paul Bingham in this issue of TDM.)

Also discussed this month, is a new and excellent font editor and print utility for the QL which I thought you should hear about. Finally, as an update to last issue's games article, a brief review of VROOM!, the grand prix motor racing game from France.

Q Draw by Psion: Q Draw is the software used by Psion to create the breathtaking graphics for QL Chess and QL Matchpoint. Very user-friendly, Q Draw can be used to create pictorial graphics of all types or to improve upon, or draw from, graphics screens produced by other programs such as GraphiQL, TechniQL, Concept 3D, Easel, or a screen created and saved from SuperBASIC. Q Draw offers four-color, high-resolution graphics as well as the more usual eight-color lower-resolution graphics. (High or lower resolution is not selectable from within the program.) Other features include a variable-width pen (or brush); the exploitation of created "shapes" which may be created or plucked from any screen stored on disk or Microdrive and then manipulated, copied, or stored for later use (a library of useful shapes is included and you may economically create your own library); two screen magnifications; and various cursor forms (a crosshair or screen grid may also be selected). There are no text or curvilinear functions which automatically create arcs, circles, or ellipses. Cursor position coordinates are not available, but this does not seem a great disadvantage in Q Draw. At $25, Q Draw is the least expensive QL graphics program and in some respects it is the best of the lot--another winner from Psion.

GraphiQL & TechniQL by Talent: These two software programs from Scotland are so complex and comprehensive that a complete description of each is impossible in an article of this length. The best I can do is describe their capabilities and differences to help you decide which of these two superb programs, offering overlapping capabilities, should suit you best.

Talent's designers have had decades of experience in designing CAD/graphics software for mainframe and minicomputer systems. Their microcomputer versions for the QL, while reducing unnecessary complexity, at the same time incorporate several never before seen features.



A sample screen from GraphiQL.

GraphiQL is a graphics design program optimized for the computer artist or illustrator. TechniQL is a two-dimensional CAD (computer-aided design) package optimized for the technical draftsman. Although the two programs have few features in common, each can do a fair job at the other's tasks with a little extra effort, but GraphiQL pictures are limited to the size of a single computer screen, while TechniQL pictures occupy many screens and be drawn in many layers (analogous to acetate overlays on an engineering drawing). Both programs have good on-screen HELP facilities. Neither has a variable-width pen (brush).

GraphiQL is primarily, a graphic arts program whose forty-six commands and other capabilities can be used for other purposes, including technical drafting. GraphiQL operates only in the eight-color, medium resolution (256 x 256 pixels) mode. Many methods of creating illustrations are possible, and cursor coordinates and other useful data may be displayed, if desired. Other features include texture and airbrush effects, screen magnification (16x), two text sizes, standard drawing shapes, and comprehensive screen/file handling. Available at $50, including a 63-page manual, GraphiQL may be found in an improved Sinclair Research version, QL Paint with icon pull-down menus and a 123-page loose-leaf manual.

TechniQL is primarily a CAD and drafting aid, but with good graphic arts capabilities (except for text-printing which is better on GraphiQL and not available in Q Draw). Additional TechniQL features include the following: four-color, high-resolution and eight-color, lower-resolution graphics selectable from within the program; about forty commands which may be executed from five pull-down menus or by two letter key codes; rapid, multi-sheet printer output; multiple magnifications over a wide range; a RAM-efficient design storage system; the capability of creating and manipulating up to 75 elements (cells) as part of a single design; and comprehensive file storage handling. At $70, TechniQL is the most expensive and comprehensive QL Graphics program.

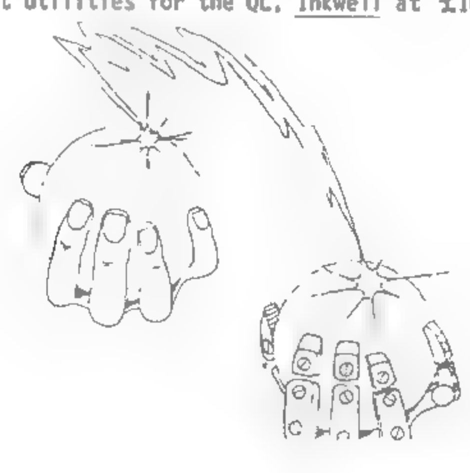
Concept 3D, distributed by an American software company located in California, is an excellent two- and three-dimensional CAD program, although the distributors do not refer to it as such. Like GraphiQL and TechniQL, Concept 3D is too comprehensive and complex to describe in detail in an article of this length.

Aptly named, Concept 3D offers several new concepts in graphic design (those familiar with Psion's VU-3D for the T/S 2068 will see some similarities). Operating in three modes, Concept 3D's capabilities may be described as excellent but with significant limitations, for example, it cannot like TechniQL produce layered designs on multiple printed sheets almost automatically. Concept 3D produces only one screen dump at a time--and that using the screen dump program on the Psion Easel cartridge.

Concept 3D is, despite its complexity, relatively user-friendly. It includes the following features... several which are unique to the QL: about 50 single- or dual-keystroke commands listed on three main menus and other sub-menus; four color, high resolution and eight-color, lower resolution graphics selectable from within the program; three types of 3D modeling, two of which are automated; rotation of objects around three axes, seen from various perspectives; image magnification and reduction over a wide range; five text sizes; hidden line removal and surface modeling; excellent documentation in a 45-page user manual.

At $40, Concept 3D is an ingenious and a well-executed program offering several features which are unique to the QL. My kind of program!

Inkwell by Palantir: An inexpensive font editor with print utilities for the QL. Inkwell at £10 (£8 to



QUANTA members) offers excellent value for money. Eight alphanumeric or symbolic fonts are made instantly available by inserting simple codes in Quill documents. Variable line spacing, print emphasis, inverse printing, and equal or proportional character spacing may be specified for symbols/font characters prepared using a 16x16 font editor. A must for desk-top publishing with the QL.

VROOM! by Pyramide: A grand prix motor-racing simulation by the distributors of 3D Wanderer. VROOM! is potentially more interesting to play than QL Hyperdrive, its QL road-racing rival, but suffers from a fault or two.

VROOM! includes five meandering racetracks of increasing complexity. Pass 10 cars and move on to the next circuit or begin all over again. Graphics and sound effects are fair to good. Your view is from the cockpit of the race-car: the steering wheel and two front wheels are seen to move in unison. Joystick steering at speeds necessary to pass cars and advance to the next circuit is very tricky.

A victim of the QDOS address differences in U.S. QLs discussed above, VROOM! does not accurately or completely depict the plan of the grand prix circuit in use. This may affect player steering: for example, while you are still shown to be on a straightaway, the track begins to curve. A second fault, perhaps related to the first, is that it is too difficult to pass another car at speed without either crashing or going off the road. At $30, $3 more than Hyperdrive, VROOM! is preferable to the former despite its faults.

NEXT ISSUE: "Optimizing QL Quill". Future articles will deal with one main topic and, typically, discuss new or related software programs.

Note: All QL programs in the article were obtained from CURRY COMPUTER, P.O. Box 5607, Glendale, AZ 85312, 602/978-2902; with the exception of Inkwell, which is available from PALANTIR PRODUCTS, Dept MF1, 60 St. Lukes Road, Bedminster, Bristol BS3 4RX, England.

Beginning Z80 Machine Code

Part Four

by Syd Wyncoop

This time, right to business! We are studying the math instructions which are listed in chart 4. This is where it starts getting a little more difficult, but not so that you can't handle it. Up to this point, most of the lessons have been peripheral background needed to make sense out of the rest of the discussion.

We only have two math functions available to us: Addition and Subtraction. As with Ld, this is not as limited as it first sounds. A study of Math Theory would teach you that all math functions are performed with addition. I'll not try to explain this further as it would fill a volume larger than all TDM's published to date. The point we need to understand and absorb is that multiplication is performed by repetitive additions. Likewise, division can be achieved by repetitive subtraction.

It is important that this makes sense to you. Think about the multiplication problem of 12X6. It can be solved by either of the following:

```
                 12
                 12
  12             12
 x 6             12
  72             12
                +12
                 72
```

Can you see how we can solve division problems by repetitive subtraction? If we had the problem 72/12, how many times can we subtract 12 from 72? Is there a remainder? Simple, isn't it?

This brings us to the first instruction...Add. We have already seen Add in operation, in Lesson 2, and probably have a good idea of its function. Trust me, it performs addition. Some of the later instructions will not be so obvious. We would read the instruction, Add A,E, as "add the value in the E register to the value in the A register and store the result in the A register".

In lesson 3, we learned that the A register is called the "Accumulator". The A register is the only register that can "accumulate" the results of eight bit arithmetic. If we had wanted the result in the E register, we would need to assign it. Can you guess the needed instruction? You get an "A" if you said Ld A,E. Otherwise, go back to lesson 3.

We also have available the instruction, Sub. The A register performs a special purpose here also. The A register is the only register we can subtract from. As with Add, the A register accumulates the result. You may see this instruction written as Sub A,C or Sub C. They mean the same thing. We will use Sub C as the A register is always implied in eight bit arithmetic.

I have mentioned several times that the A register will accumulate the results of eight bit arithmetic. We need to leave the instructions for some more background.

We have already learned that a single register may only contain a value in the range 0-255. There is a condition known as an "overflow" which occurs when these values are exceeded. The simplest way to describe overflow is by example. Let's assume we are adding 255+1. We have not discussed number systems yet (that's a later lesson) but let's show our example in binary as it will demonstrate the point dramatically:

```
Decimal       Binary
  255        11111111
 +  1       +       1
  256      1 00000000
```

Look closely at the binary example. Each digit represents a bit of the A register (or any other eight bit location). Assume for now that my answer is correct, and you will note that we are now trying to place a nine bit number into an eight bit hole! The answer returned in this case would be 0, instead of the expected answer of 256. Our example shows an eight bit overflow, but can you see how we overflow a register pair (sixteen bits)?

Our friend, the CPU, has a special register, F, which we learned stands for Flag. It is called this because its job is to keep track of various things for the CPU. This is accomplished by the setting or re-setting of a bit of the F register. Setting a bit makes it a 1, and re-setting it makes it a 0. We will discuss this in some detail at a later time.

The bits are referred to as flags due to the fact that they indicate wether or not a certain condition exists. The flag we are now interested in, is the Carry

flag. We will also discuss the F register later, therefore, we only need to consider the Carry flag now.

In the above example, we found we would receive an answer of 0. The ninth digit is not lost, as it is placed in the F register as the carry flag. In other words, the Carry flag takes on the value (either 1 or 0) of the overflow from out arithmetic operation. We will soon wee why we would want to save the carry.

Back to the math instructions. We have available the instruction ADC which is read add with carry. To see the difference, another example:

```
Add A,E   means  Let A=A+E
ADC A,E   means  Let A=A+E+Carry (keeping in
                 mind that the carry will again
                 be set or reset by the result)
```

ADC will allow us to chain together the needed additions to guarantee the correct result. Some of the same results can be achieved with the register pair instructions, however, there can still be overflows. Study the following to see what I mean:

```
Ld HL,0040h         Ld H,00h
Ld BC,7FFFh         Ld L,40h
Add HL,BC           Ld B,FFh
Ld B,H              Ld c,7fh
Ld C,L              Ld A,L
Ret                 Add A,C
                    Ld L,A
                    Ld A,H
                    ADC A,B
                    Ld B,A
                    Ld C,L
                    Ret
```

Both of these routines will do the same job. Which makes more sense? Uses less memory? Executes faster? The point is that there are many ways to get the job done and many considerations to why we should choose one over another.

We also have SBC or subtract with carry. This one is special because it is the only way to perform sixteen bit subtraction. We cannot Sub HL,BC. We must SBC HL,BC which implies we know the status of the carry flag. We may not know what's on carry's mind, but we can clear the carry flag prior to performing a SBC by doing an addition, that we know will not generate a carry. One that will work in all cases is Add A,0. The value of A is unchanged and the carry flag is reset (0) or cleared as there is no overflow. We will find other ways to clear carry, soon.

We need to be aware that HL acts as the accumulator for sixteen bit arithmetic operations. HL has much the same favorite status with the CPU as does A. The reason we need an eight and a sixteen bit accumulator is that we cannot add or subtract registers from register pairs and vice versa. In other words, we cannot Add HL,A.

The last instructions for this lesson are special cases of Add and Sub. They are Inc and Dec which are short for increment and decrement. Each will Inc or Dec by one. For example:

```
Inc HL    [redacted]  Let HL=HL+1
Dec HL    means       Let HL=HL-1
```

Armed with these new instructions, see if you can rewrite the addition routine we had in lesson 2, to avoid the overflow error it contains. Make sure the last instruction is a Ret and use PRINT USR address to run it and return the answer to BASIC. See if you can write a similar routine to perform subtraction.

A final note on the charts I am providing. This is the last time I will include the abbreviations comments. Also, you can usually substitue IX or IY for HL and (IX+d) or (IY+d) for (HL). Therefore, I will not include them in the charts.

Until next time...happy computin'.

Chart A

Registers	Register Pairs
Add A,r	Add HL,rr
Add A,n	Add HL,SP
Add A,(HL)	Add IX,rr
Add A,(IX+d)	Add IX,SP
Add A,(IY+d)	Add IY,rr
	Add IY,SP
ADC A,r	
ADC A,n	ADC HL,rr
ADC A,(HL)	ADC HL,SP
Sub r	SBC HL,rr
Sub n	SBC HL,SP
Sub (HL)	
	Inc rr
SBC A,r	Inc SP
SBC A,n	
SBC A,(HL)	Dec rr
	Dec Sp
Inc r	
Inc (HL)	
Dec r	
Dec (HL)	

Where: r =any single register
rr=any register pair
n =any numeric constant 0-255
nn=any numeric constant 0-65535
d =any displacement 0-255
pq=any address

TS 1000/1500 PROGRAM CHAINING
Part Three

by Earl V. Dunnington

Parts One and Two of this series covered the VARS, System Variables, and the Safe Area methods of passing data from one program module to another in a chained program.

The Above RAMTOP method of passing data, in chained programs, is very similar to the Safe Area method and is the best of all of the methods, as data stored above RAMTOP is protected from LOAD, RUN, NEW, an expanding program, or the expansion of the display file. About the only thing that can wipe out data properly stored above RAMTOP is a program crash, a power failure, a program bug, or resetting RAMTOP.

The amount of bytes or addresses you can lower RAMTOP and still have the program RUN is determined by the Upper and Lower Limits of the Safe Area of the program. In a Chained Program, the module that requires the most memory in order to RUN, determines the address to which RAMTOP can be set for the entire program. A method for finding the Upper and Lower Limits of the Safe Area and the minimum setting for RAMTOP that will allow the program to run, was presented in the series of articles: "Adventures In The RAM Jungle And Other Mysteries" (see Sept/Oct '85 thru Jan/Feb '86 issues of TDM).

When the computer is turned on, the address of RAMTOP is the first nonexistent byte at the top of the user

available Random Access Memory (RAM). For the 1k ZX81, this address will be 17408; for the 2k TS 1000 it will be 18432; and for the 16k TS 1500 it will be 32768. When a 16k Ram pack (TS 1016) is attached and the computer turned on, RAMTOP is at address 32768 for all three computers. To check the address of RAMTOP, ENTER:

```
PRINT PEEK 16388+256*PEEK 16389
```

This only returns the value stored in the system variable RAMTOP. To check that RAMTOP is actually at this address, let A= the address stored in the system variable. Then ENTER:

```
PRINT PEEK (A-1)
```

The result should be 62

In the case of the TS 1500 with the 16k Ram Pack attached, the bytes from address 32768 to 49151 are above RAMTOP and can be used for storage of data including machine code programs. Any part or all of the additional memory can be incorporated into the BASIC programming area by raising RAMTOP. Of course RAMTOP can also be lowered.

Should you have an odd amount of RAM, to find the maximum address to set RAMTOP, add to 16 the k of the RAM and multiply by 1024 (the number of bytes in one k). For example, if you have four k RAM:

(16+4)*1024=20480

However, the maximum address that you could set RAMTOP is 65535 not 65536 as the maximum value you can POKE into an address is 255.

RAMTOP can be lowered to make room for the storage of data by POKEing the address desired into the system variable RAMTOP and then entering NEW. The system variable RAMTOP consists of two bytes located at addresses 16388 and 16389. The formulas for POKEing the low byte into the lower addresses and high byte into the higher addresses are given on page 134 of the TS 1000 and page 160 of the TS 1500 User Manuals. For example, to set RAMTOP to 18000, then n=16388 and v=18000. Substituting in the formulas, type into the computer:

```
10 POKE 16388,18000-256*INT (1
8000/256)

20 POKE 16388+1,INT (18000/256
)

30 NEW
```

Now RUN the program. To check that RAMTOP was moved, ENTER:

```
PRINT PEEK 17999
```

The result will be 62. As you can see, the NEW command wipes out the program. It also destroys any variables, strings and dimensions. This precludes the use of this method of setting RAMTOP in a module designed to automatically LOAD the next module of a chained program. A routine for setting RAMTOP without destroying the program or any variables or strings was presented on pages 9 and 10 of the July/August '85 issue.

Although RAMTOP cannot be set using NEW in a module designed to LOAD the next module, it is used in some types of programs where the operator does the linking. NEW is also used where machine code is to be shifted above RAMTOP to wipe out the program in order that a new program be typed into the computer. This can be accomplished by POKEing the code into the Safe Area for the program, before the line containing NEW. For example, clear the computer memory by turning it off and then back on. Type in the program given above adding the following line:

```
25 POKE 18001,255
```

RUN the program. Now ENTER the direct command:

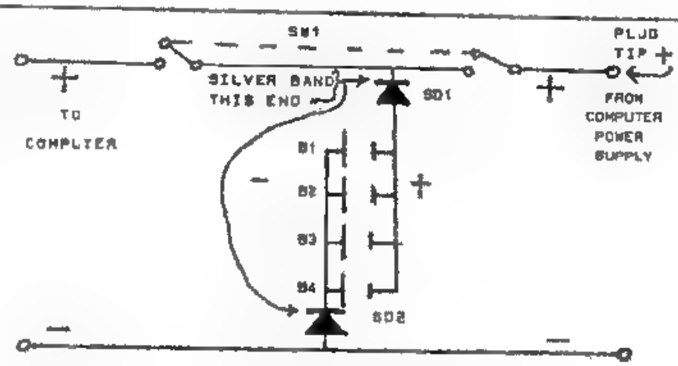
```
PRINT PEEK 18001
```

As you can see the 255 is still in address 18001, RAMTOP is set below this at 18000, and the program has been destroyed.

For the 1k RAM ZX81: Set RAMTOP to 17000 using v= 17000, PEEK 16999 for the 62, change Line 25 for address 17001, and PEEK 17001 for the 255.

If you intend to do serious programming, work with chained programs or use word processors, then you should have two accessories besides an interfaced printer. The first is a tape recorder (or other storage device) capable of consistant SAVEs and LOADs. Two is even better, one for LOADing programs or data connected Ear to Ear with the computer, the other for SAVEing programs or data connected Mic to Mic. I use a Radio Shack CTR-51 only for LOADing as it has a wider allowable volume control setting range on some commercially produced software. My other tape recorder is a GE Computer Data Recorder model no. 3-5158B.

The second accessory is an emergency power supply particularly in Florida where we refer to the power company as "Florida Flicker and FLash". Figure No.1 is a circuit diagram for an automatic emergency power supply that will maintain the program for power interuptions up to 15 minutes. You must remember to throw the switch off before disconnecting the regular computer power supply or before plugging into the computer.



B1,B2,B3,B4-9 VOLT RECHARGEABLE BATTERIES
SD1,SD2-SILICON DIODES 1N4001 RADIO SHACK 276-1101
SW1-DPDT SWITCH RADIO SHACK
FIGURE NO. 1

In addition to the parts listed on the diagram, you will need the following items:

1 project box large enough to accomodate the circuit plus the four batteries.

1 rubber grommet to protect the leads to the computer.

4 nine volt battery connectors.

1 two conductor 1/8" modular phone jack, open circuit type, panel mounting (Radio Shack 274-251).

1 two conductor 1/8" mini phone plug (Radio Shack 274-286).

The use of a 12 volt latern battery instead of rechargeable 9 volt batteries and a battery powered tape recorder, would allow the SAVEing of the program or data. However, leaving the switch on inadvertantly without the computer power supply on could be costly. If you use the 12 volt battery, also use 2.5 amp silicon diodes (Radio Shack 276-1114). As diodes are easily damaged with heat...heat sink them while soldering.

A practical chained program illustrating the Above RAMTOP method of passing data will be presented in Part Four of this series of articles.

MARKEL ENTERPRISES

A+ Computer Response Authorized Dealer

Post Office Box 2392
Secaucus, New Jersy O7O94-O992
(718) 627-1293

Serving the Sinclair community since 1982!

QL Computer w/software & free ADDRESS BOOK software	$289.95
QL Vision RGB color monitor (w/cable) while supply lasts	$299.95
QL Near Letter Quality printer (w/cable)	$289.95
QL Computer w/monitor -or- printer	$575.OO
QL Computer w/monitor -AND- printer	$795.OO
512K RAM expansion w/thruport	$220.OO
QL technical guide	$29.95
Microdrive Cartridges - book of 4	$10.95
RS232 Printer Cable	$14.95
Modapter - for use with all RS232C modems - w/software	$54.95
QL Monitor	$39.95
QL Touch and Go Typing Tutor	$25.95
QL Gardener	$29.95
QL Decision Maker	$49.95
QL Project Planner	$49.95
QL Entrepeneur	$49.95
QL Assembler	$49.95
Cartridge Doctor - restore lost files	$23.95
Icon Controlled Environment	$39.95
CHOice cartridge (multitasking/mailmerge/ramdisk) for I.C.E.	$24.95
QSPELL spelling checker for Quill	$34.95
QDUMPS screen dumps to Epson compatible printers	$15.95
Concept 3D computer aided design	$39.95
Mail Merge	$24.95
QL Peintre graphics program	$34.95
Nucleon toolkit	$39.95
QL Caverns	$15.95
QL Chess	$27.95
QL Matchpoint	$24.95
QL Bounder	$29.95
QL Pawn	$36.95
Wanderer (must have RGB monitor for this true 3D game)	$37.95
VROOM	$29.95
Karate	$29.95
War in the East - Master Module & Secenario I	$29.95
Master Module & Scenarios I & II	$44.95
Master Module & Scenarios I, II & III	$59.95
Individual Scenarios	$19.95

PASCAL, FORTRAN AND OTHER LANGUAGES AVAILABLE...WRITE FOR DETAILS

FREE I.C.E. CARTRIDGE WITH EACH QL ORDERED BEFORE DECEMBER 25, 1986

——STUDENT DISCOUNTS—

MORE ITEMS BEING ADDED DAILY AS WE EXPAND OUR QL SUPPORT.
IF IT'S NOT LISTED – WE CAN GET IT – PLEASE CALL OR WRITE.

ADD 3% FOR SHIPPING AND HANDLING.
NY AND NJ RESIDENTS ADD APPROPRIATE SALES TAX.

Our address says mail order / Our phone says NYC / Our name says support

Understanding And Upgrading The TS1016 RAM Pack

by Tim Stoddard

This is the first of a two part article on how dynamic rams operate, how the TS1016 works, and how to upgrade the ram pack to use the newer 5 volt-only 64k dynamic rams.

First let me put to rest all the fears that are probably running through your minds right now! Dynamic rams have had a bum rap for years. They are not only EASY to use but they are also much easier to wire up! Have you wired, or can you imagine wiring up, an array of 64k memory using static random access memory? Even if you used the now inexpensive 6116 CMOS 2k byte-wide rams you would need 32 of them, plus the supporting selection logic on a board that will barely fit in an S100 system! If you used the affordably latest in CMOS ram (8k byte-wide), then the resulting 8 28-pin chips plus selection logic would fit on a board about the same size as your TS1000/ZX81 computer!

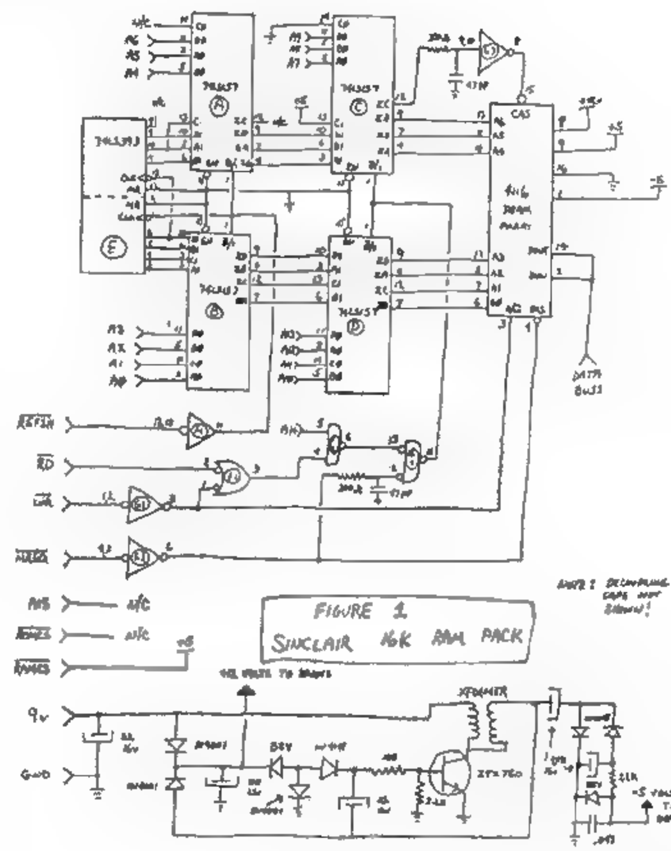
Dynamic rams, or DRAMs for short, are constructed of simply one transistor and one capacitor per data bit. Static Rams, or SRAMs for short, are constructed of a flip-flop consisting of 4 or more transistors per bit. Right off you can see that the SRAMs consume much more power and much more die space (die is the actual silicon chip that the DRAM is made of) than DRAMs. DRAMs are not without fault, however, in that they require a small amount of attention to timing, and refresh to use.

With SRAMs you simply supply an address and wait the required access time for the data to appear at the output. DRAMs, on the other hand, require multiplexing of the address bus. Why? Take a look at a typical DRAM such as the one in your TS1016 ram pack and count the leads on the IC body. I count 16. Well, lets see...the 4116 rams used in the TS1016 ram pack need 3 supply voltages and ground (+5, +12, -5, GND)...that leaves us 12 leads for address and control; or for the 64k DRAM, 2 leads are used for power (+5, GND), so that would leave 14 pins for address and control. Now let's supply the 14 address lines needed for 16k, or for the 64k DRAM 16 address lines...oops, we just ran out of pins.

The manufacturers came up with a scheme for cramming 14 lines into the 7 that are needed for the 16k DRAM, or 16 lines into 8 needed for the 64k DRAM by multiplexing them: First, you supply the lower 7 or 8 address bits to the address lines and strobe them into the DRAM. The DRAM contains a Row address latch that holds these 7 or 8 address bits, and in fact, the strobe line used to strobe them in, is called RAS (Row Address Strobe). Next, we switch to the upper 7 or 8 address lines via a TTL multiplexer switch and then strobe in the column address using a second strobe line on the DRAM chip called CAS (Column Address Strobe). Lastly, we wait the required access time and then read the data out of the DRAM. The only other requirement we must observe is the refresh timing needed by the DRAMs.

Since each bit in a DRAM consists of just one transistor and a capacitor, it is easy to see that there is no stable state like that in flip-flop type memory found in SRAMs. The capacitor soon starts to lose its charge via leakages of one type or another, and before you know it you've lost the state that was stored in that capacitor. In order not to lose the capacitor's state we must periodically refresh each capacitor to preserve its current state. Refresh simply means that we want to preserve whatever state the storage capacitor is currently in, the two states being charged for a logic high, or discharged for a logic low.

The manufacturers determined that if each capacitor was refreshed within a specific amount of time, enough of its charge would still be there to determine what



its current state should be. For most 16k DRAMs such as those in the TS1016 ram pack, and the newer 64k DRAMs, each location must be refreshed every 2 milliseconds.

Reading all 16,384 locations to refresh the DRAM, would a considerable amount of time. Another way to refresh the DRAM is needed to keep the refresh time down. One way takes into account the fact that when a row is addressed, that ENTIRE ROW is refreshed! So if we address just 128 row locations, the entire 16k would have been refreshed. I should also point out at this time that the 64k drams are internally arranged so that they only need 128 row refreshes to refresh the entire DRAM. THey are internally set up as four 16k blocks. Since the row addresses are supplied to each of the four 16k blocks at once, it follows that only 128 refresh cycles will refresh the entire DRAM.

A method that takes advantage of an entire row being refreshed while applying the row address is called RAS only refresh. It is enabled by supplying the refresh address to the address pins of the DRAM and then enabling the RAS line ONLY. THis will refresh the entire row addressed by the address pins. Also, since we do not supply the column address and the CAS signal, the DRAM will not complete a true read operation, and therefore, will not output any data, but will remain in a tri-state condition.

There are a number of other refresh modes, especially in the newer 64k and 256k DRAMs, and if there is enough interest in this article, I'll describe those modes in a future article. For now, let's proceed to the inter-workings of the TS1016 ram pack with this new knowledge of DRAMs under our belts.

I could not procure a schematic of the ram pack, so I dissected a ram pack that I own and drew a schematic

from that. During the following discussion, please use the schematic in figure 1, and the timing relationships in figure 2.

The first thing you Z80 hackers will notice is that there is a refresh counter in the ram pack. Anyone who has worked with the Z80 knows that it has its own refresh counter on chip, so why use an external one? The "R" register, as it is called in the Z80, is used in the display interupt routine to count the number of characters per line, and since the "R" register is manipulated alot in this routine, it would not be wise to use this register to preserve your data.

ICs "A" and "B" are 74LS157s (quad 2 line to one line multiplexers) and are used to switch in the refresh address counter IC "E", a 74LS393 dual binary counter. The counter is needed to "remember" what address we need to refresh. The refresh request signal is supplied by the Z80 CPU in your TS/ZX computer and is called REFSH on the schematic. So when REFSH (active low) comes into the ram pack it gets buffered and inverted by gate "F4" and is then sent to pin 1 of both ICs "A" and "B" which will cause them to select the inputs suffixed with a "1" (A1,B1,C1, and D1). THose inputs come from the outputs of the 74LS 393 refresh address counter. Note also that the same line used to select the refresh counter will also advance the counter one count AT THE END OF THE CURRENT REFRESH CYCLE. The counter will now contain the next address needed to refresh. ICs "C" and "D", also multiplexers, are switched by gate "G4" which is enabled by gate "F1" via gate "F2". During a refresh cycle there are no active RD or WR signals from the Z80, so the output of gate "F1" is low, the output of gate "F2" then is high, and the output of IC "G4" is also high. The multiplexers "C" and "D" therefore, take input via the inputs suffixed with a "1" (A1,B1,C1, and D1), which is from the refresh counter via the multiplexers "A" and "B". The DRAM address lines are at this point "connected" to the refresh counter IC "E" via the four multiplexers.

The last requirement to refresh the DRAM is to supply the RAS signal. This is accomplished with the MREQ signal supplied by the Z80 CPU. The MREQ signal is buffered by gate "G2" and supplied directly to the RAS input of the DRAM. This low-going signal latches the refresh address in the DRAM causing that entire row to refresh. Note that since gate "G4" is disqualified by the output from gate "F2", which we discussed earlier, its output will never change during the refresh cycle. This will keep the refresh address supplied to the DRAM and also keep the CAS line to the DRAM inactive (high) throughout the refresh cycle, thus preventing a true read operation of the DRAM. The output of the DRAM, therefore, during refresh is at its tri-state condition. The MREQ signal now goes inactive (high) which removes the RAS signal from the DRAM ending the refresh cycle. Finally, the inverted REFSH signal coming out of gate "F4" now goes low, switching the multiplexers "A" and "B" back to the system address lines, and at the same time the low-going signal advances the refresh counter IC "E" via pin 1 to the next row address in preparation for the next refresh cycle. That wasn't so bad was it? Now for a RD/WR cycle.

First, keep in mind that because the REFSH signal is inactive during a read or write cycle, the multiplexers "A" and "B" are supplying address lines A0 through A6 to the inputs suffixed with a "1" on multiplexers "C" and "D". Multiplexers "C" and "D" now select either system address lines A0 through A6, or system address lines A7 through A13.

The first signal to occur after the system has supplied the proper address is the MREQ signal. This signal, after passing through buffer "G2", is supplied to the RAS pin of the DRAM. At this instant, when the RAS signal has just gone active (low), the DRAM is "looking" at system address lines A0-A6, and this address gets latched into the DRAM's internal RAS latch. MREQ is also supplied to gate "G4" via a time delay net-

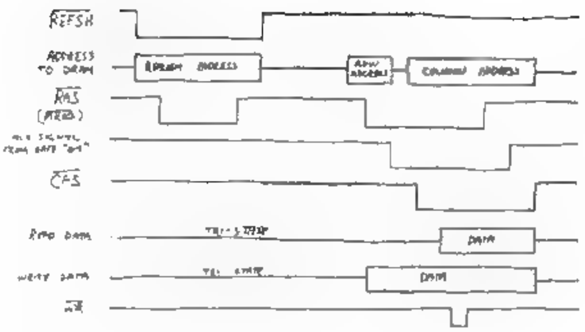


FIGURE 2

work consisting of a 300 ohm resistor and a 47 PF cap.; and befor the MREQ signal has had time to get through the time delay circuit, the output of gate "G4" is at its inactive (high) state. This keeps multiplexers "C" and "D" selecting address lines A0-A6 via multiplexers "A" and "B". About 50 to 100 nanoseconds later the MREQ signal finally gets through the time delay circuit and partially enables gate "G4". The other leg of gate "G4" is enabled for a read or write operation (which we are doing) and system address line A14.

System address line A14, when in its high state, is used to select the ram pack by placing it in the 16k to 32k system address range. Note that since the last address line A15 is not defined anywhere, a mirror image of the ram pack will be found in the 48k to 64k area of system ram. Anyway, gate "G4" is now active and its output goes active (low). This signal now switches the multiplexers "C" and "D" to the system address lines A7-A13. Note, too, that for multiplexer "C" the output ZC goes from high to low via inputs C0 and C1. This will supply a low going signal to a second time delay circuit. Why? We have just switched the address lines to the DRAM and before latching the address in the DRAM, we must now allow some time for the multiplexer output to "settle" before enabling the CAS signal to the DRAM, this time is given to us with the second time delay circuit. After 50 to 100 nanoseconds the signal gets thru the time delay circuit and is supplied to the CAS pin on the DRAM. This latches address lines A7-A13 into the CAS latch in the DRAM. About 50 nanoseconds later, the DRAM will supply its output data via pin 14 to the system data buss during a read cycle, or for a write, it will strobe in the data from the data buss on the falling edge of the Z80 supplied WR signal.

The only way the DRAM knows what type of cycle the Z80 is in, is via pin 3 on the DRAM. When low, it is a write cycle, and when high, a read cycle. This signal is supplied by the Z80 CPU and occurs during the MREQ signal. After the Z80 has read or written the data, it will make the MREQ signal inactive (high), this will then make RAS inactive (high), also causing gate "G4" to switch multiplexers "C" and "D" back to inputs A1-D1 thus causing CAS to go inactive, tri-stating the DRAMs data output and ending the read/write cycle.

The one transistor circuit at the bottom of the schematic is a DC to DC converter that supplies the needed +12 and -5 volt bias supplies for the 4116 DRAMs. This circuit is a source of a lot of noise and will be eradicated when we upgrade the ram pack.

I know that the above discussion is somewhat "dry", but if you can come to understand what is going on, you'll be a long way into understanding what makes your computer "tick"!

In the next issue, I'll present the needed modifications to upgrade the TS1016 to 64k, plus some of the unusual restrictions imposed by the TS1000/ZX81 architecture in designing 64k ram systems.

CLASSIFIED

FREE ADS FOR SUBSCRIBERS

ADVENTURERS! Join in the fastest growing hint trading group in North America. Send SASE with hints, and problems, for prompt reply to: Douglas Jeffery, Larch Rd., RR #1 Telkwa, B.C., Canada V0J 2X0.

WANTED: Programs for TS-1000 to trade. Send me your list and large SASE. Also programs for above 16K RAM. Anyone have a copy of Memotech Memopak 64K Memory Instruction Manual? Any information will be appreciated. Richard Beier, One Darwin Drive, N. Merick, NY 11566.

AERCO C/P $50; TS-1000 with power supply and man. $10; 2068 Software $35. D. Gustafson, P.O. Box 3700, Pens. FL 32516 (OS-64, ML Tutor, VU-3D, MSCRIPT/disc).

SOFTWARE for TS-2068, TS-1000, and AERCO FD-68. New and used hardware/ software. SASE for free catalog. Chia-Chi Chao, 73 Sullivan Dr., Moraga, CA 94556.

WOULD LIKE TO CONTACT another Zebra Graphics Tablet User. Need more info on Interface and Software--Alvin Lam, 1258 Fascination Circle, Richmond, CA 94803.

COMING SOON. MONEY MACHINE II. Now starring VANNA WHITE. Send SASE for FREE LLISTing of Vanna White moving graphic. Herb Bowers, 2588 Woodshire Cr., Chesapeake, VA 23323.

SINCLAIR QL COMPUTER and extra software, $275 used. Delta 3-in-1 interface with 128K, a disk interface and parallel printer port, $225. Two 3" Amdek disk drives, power supply and cable, $125. ALL for $550 and shipping. 904/378-9000 evenings.

FOR SALE: extra copy of 352-page QL ADVANCED USER GUIDE by Adrian Dickens--THE authoritative guide to the QL & QDOS. Asking $20 ppd (compare at $29.95) send to Paul Bingham, POB 2034, Mesa, AZ 85204.

T/S 2068 UTILITIES: Auto-line Number, Renumber (GOTOs, GOSUBs, ect.), Read Tape Header, Display Vars., Find String in Prog. $10. Michael Cover, 1405 Chevington Cir., Zanesville, OH 43701.

FOR SALE: Over 30 game programs for T/S 2068 or Spectrum at $8.00 (U.S.). Send for a list of software available or phone (403) 236-4465. Garvin Bertrand, #311 135 Lynnview Rd. S.E., Calgary, Alberta Canada T2C 2G6.

---

6 TS BOOKS FOR SALE: Working 2068, Creative Games For 2068, Games For 2068, TS 2068 Explored, The 2068, 200 Computer Programs In BASIC For 1000 + 2068. C. Lytle, 1064 Worth. Woods, Worth., OH 43085.

T/S 1000, 2050 MODEM, Millenia K Disk I/F, Zebra Lightpen, lots of S/W: 615-823-2078 p.m. only. D. Taylor, Rt 4 Box 303-A, Livingston, TN 38570 for titles, prices, more info.

FOR SALE: 2068 w/Tech Manual, Alphacom 32 printer, 2 programs & 3 texts. ZX81, Winkee & 6 texts. 100 hrs on each. S.L. Jacobs, Rt 3 Box 2750, Dexter, ME 04930, 207-924-6414

Do you have some equipment or a program that you would like to sell? Looking for something hard to find? Place an ad in **THE CLASSIFIEDS!** Subscribers can place one free personal ad in each issue. Ad size is 32 Col. wide (like 2040 paper) and maximum of six lines. For additional lines - $3 each. **NON-SUBSCRIBERS** and **DEALERS:** $4 a line. **DEADLINE FOR ALL CLASSIFIED ADS:** Two weeks before publication date. Mail your ad to: **TIME DESIGNS MAGAZINE, The Classifieds Dept., 29722 Hult Rd., Colton, Oregon 97017.**

WANTED TO BUY: Ramex Millennia K Disk Drive System w/MPI 1-meg drive. Also same drive #2 for above. Send price quote to- B. Hardware c/o Jim, 132 1st Ave. S., Perham, MN 56573. 218-346-4760.

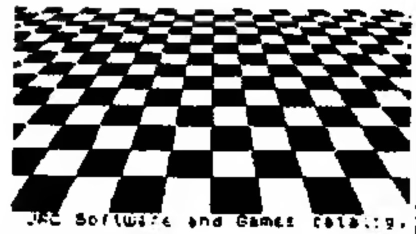
WANTED: Screen dump program on cassette to drive Tandy CGP-220 Color Ink Jet Printer for TS-2068 w/ Tasman B I/F and/or on mcdrive for QL. Michael Kudelka, 4859 Sacramento, St Louis, MO 63115.

FOR T/S 2068: 3-Reel 9-Way Win Giant Slot Machine on cassette. All in BASIC & listable, about 32K with Stick/Inkeys provision. Send $6.00 to D. JEWETT, 63 Charro Dr., Santa Rosa, CA 95401. Includes shipping.

WOULD LIKE TO CONTACT MIDI I/F OWNERS who use 2068 or Spectrum and any keyboard. Professional musicians or amateur hobbyists. May start Sinclair MIDI users group/newsletter. Write to: Tim Woods c/o TIME DESIGNS, 29722 Hult Rd., Colton, OR 97017

ORDER OUR CATALOG!

and get this *FREE* Software on cassette...



FOR THE TS2068

1. 3D Moving Display
2. 3D Pyramid
3. DIAMOND MIKE game demo
4. 3D Tic-Tac-Toe demo
5. Plus commentary by programmer John Coffey on 32-bit computers.

Our unique product catalog and free software all on a quality tape only...

$1.00

Order Yours Today!

JRC Software
P.O. Box 448
Scottsburg, IN 47170

QL MULTI-MICRO SPECIAL: $288 WITH FREE EXTRA SOFTWARE!

Get an Extra FREEWARE cartridge Filled with additional Super Software with ALL QL purchases right NOW! Get Backup Utilities, Graphics Demos, Clocks, Maps, Sound Generators, a Postermaker! Too Much to mention! All Free with each QL ordered. Add $7.00 for S&H please. ALSO: SPECIAL PROGRAMMERS, OS/LANGUAGES, EDUCATIONAL & BUSINESS SYSTEM PACKAGES! MODEMS. CALL us up on our Q Line!

But what IF you Already HAVE your QL?? THEN Just send in a blank cartridge with $2.00 for S&H and you'll get all of the above FREE Programs Too! We offer users QUANTUM SUPPORT for the QL! The Computer That By Far Outclasses the Status Quo PCXTAT & Cloneys!

QUANTUM COMPUTING, Box 1280, Dover, N.J. 07801 CALL: (201) 328-8846 The QLINE - ANYTIME 24 HOURS!

"Greetings human, I am H.E.N.R.Y., winner of the Golden Droid Award at the First International Personal Robot Congress and Expo in 1984. You may be surprised to hear that my brain is really a Timex/Sinclair 1000 computer. In fact, you can learn more about me and other projects in a new book by my master, Bruce C. Taylor. The book is called: BUILD A MICROCOMPUTER-CONTROLLED ROBOT. It has over 200 pages, over 80 illustrations, and comes with a great looking custom 3-ring binder. There is even a bonus section for instructors who would like to teach about robotics. You can get your very own copy for only $17.00 (postage paid) from the nice folks at TIME DESIGNS MAGAZINE COMPANY. Get a head start on personal robotics...the newest high-tech craze. Get my book, and read all about me...how could you go wrong human?"

Quality Publications For Sinclair Computers

TIME DESIGNS MAGAZINE COMPANY

29722 Hult Rd. • Colton, Oregon 97017

TS2068 Trackball Only $19.95

Originally sold for $69.95
Specify Cat# TBTMX02

Plugs into TS2068 Joystick Port and works with all joystick software.

Bonus Feature: Also works on Commodore 64, VIC-20, ATARI 800, and more. Contact factory for more complete list.

You can benefit from our recent purchase of brand new WICO Trackball Controllers at closeout prices. We've taken the model WICO originally made for the Texas Instrument 99/4A and made a very simple modification so it now is fully compatible with the Timex TS2068's joystick port.

WICO is the largest designer and manufacturer of control devices for commercial arcade video games. If you've ever played an arcade video game, chances are you've used a WICO joystick or trackball. You've experienced the superior control. The pinpoint firing accuracy. The exceptional durability.

Features: Phoenolic ball offers 360-degree movement. Two optical encoders provide split-second movement. Quick-action fire button for smooth, two handed arcade response and feel. Long 5' computer connection. Heavy duty plastic case for long hard use.

The WICO warranty has been voided by our modification. But we give you our 15-day money back guarantee and a one-year limited warranty from Zebra Systems.

Timex Games $2 Each

With your order for a TS2068 trackball you can purchase any of the following Timex TS2068 Trackball and Joystick compatible games at the special low price of $2.00 each for cassettes and $3.00 for cartridges.

CAT#	TITLE
	Cassettes at $2.00 each
64001	Androids
64002	Penetrator
64004	Casino I
64005	Crossfire
64006	Circuit Board Scramble
64007	Dragmaster
64009	Guardian
64012	Fun Golf

CAT#	TITLE
64014	Hungry Horace
64015	Horace Goes Skiing
64019	Horace and the Spiders
64021	Blind Alley
64023	Crazybugs
	Cartridges at 3.00 each
74001	Androids
74005	Crazybugs

$5 Off Tech-Draw Jr.

You can save $5.00 on the purchase of Tech-Draw Jr. if you purchase it at the same time as a TS2068 trackball. Instead of the regular price of 19.95 you can get it for 14.95. See our catalog for a complete description of Tech-Draw Jr. and a list of printers that it supports. Order Tech-draw Jr. Catalog# C256.

TS1000 TRACKBALL Only $39.95

Originally sold for $109.95
Specify Cat# TBTMX01

Plugs into the back of TS1000,1500,2068, or ZX81.

We've taken WICO's Apple II trackball and put its controller card on an interface adapter for the Timex bus. Now you can get all the benefits of the Apple Trackball with its intelligent controller card, on your Timex computer.

The Apple trackball controller has sixteen integrated circuits on it that read the optical electronic encoder wheels from the trackball and completely keep track of the trackball movement with separate x and y direction up/down counters. This enables your Timex computer to get the position of the trackball by just reading two input ports. This is a tremendous advantage on the TS1000, since the computer can be left in "SLOW" mode for smooth graphics, while the trackball interface card does all the work.

The Apple II trackballs alone originally sold for over one hundred dollars. Now you can take advantage of Zebra's recent purchase of a large number of them at closeout prices. You get the Apple II trackball with Apple interface card, Zebra's Timex- to-Apple bus adapter, and complete instruction manual with sample routines for all the Timex computers. And all for just $39.95.

Ordering Instructions: Include $3.00 S&H for UPS. P.O Boxes and other orders requiring U.S. Mail must add $4.00 extra shipping per trackball. VISA/MC Accepted. NY Residents add sales tax. Order now! Quantites are limited to stock on hand.

Zebra Systems, Inc.

78-06 Jamaica Ave.
Woodhaven, NY 11421
(718) 296-2385
HOURS: M-F 9AM-5PM

TRACKBALLS FOR OTHER COMPUTERS

We have bargain priced WICO trackballs adapted to just about every popular computer on the market. Send a SASE or call for a complete list.